
Netze

Modul 12: TLS und VPN

👤 Prof. Dr. Hannes Tschofenig



18. Dezember 2025

Semesterplanung — Vorlesungen

Modul	Dozent	Datum	Thema
1	Rademacher	2. Oktober 2025	Einführung, OSI-Referenzmodell und Topologien
2	Rademacher	9. Oktober 2025	Übertragungsmedien und Verkabelung
3	Rademacher	16. Oktober 2025	Ethernet und WLAN
4	Tschofenig	23. Oktober 2025	IPv4, Subnetze, ARP, ICMP
5	Tschofenig	30. Oktober 2025	IPv6 und Autokonfiguration
6	Tschofenig	6. November 2025	Netzwerksegmentierung
7	Tschofenig	13. November 2025	Routing
8	Rademacher	20. November 2025	Transportschicht und UDP
9	Rademacher	27. November 2025	TCP
10	Rademacher	4. Dezember 2025	DNS und HTTP 1
11	Tschofenig	11. Dezember 2025	HTTP 2 und QUIC
12	Tschofenig	18. Dezember 2025	TLS und VPN
/	/	8. Januar 2026	Bei Bedarf / TBA
13	Tschofenig	15. Januar 2026	Messaging
14	Rademacher	22. Januar 2026	Moderne Netzstrukturen



Semesterplanung — Übungen und Praktika

ID	KW	Art	Thema
	40	/	/
UE-1	41	Übung	Topologien und OSI
UE-2	42	Übung	Übertragungen bspw. Kabel
P-1	43	Praktikum	Laboreinführung und Netzwerktools
S-1	44	Video	IPv4
P-2	45	Praktikum	Adressierung
P-3	46	Praktikum	IPv4 und Autokonfiguration
P-4	47	Praktikum	IPv6 und Autokonfiguration
P-5	48	Praktikum	Routing
P-6	49	Praktikum	Switching
P-7	50	Praktikum	Transportprotokolle
S-2	51	Experiment	VPN
S-2	52	Experiment	VPN
	2	/	/
P-8	3	Praktikum	DNS
P-9	4	Praktikum	Webkommunikation

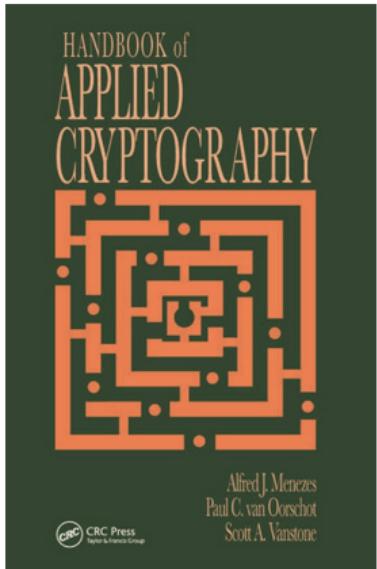
UE - Übung laut Stundenplan in den Seminarräumen

P - Praktikum in C055

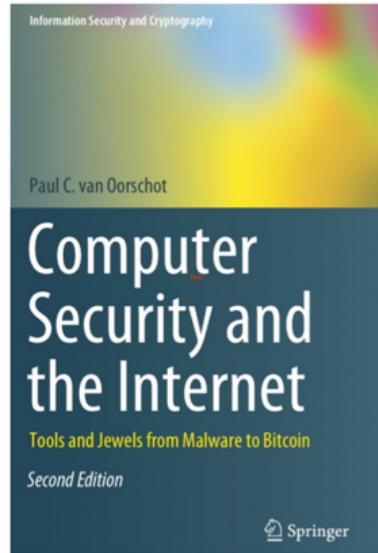
S - Selbststudium KEINE Präsenz

Agenda

- Was versuchen wir zu schützen?
- Bausteine
 - Symmetrische Verschlüsselung und Entschlüsselung
 - Hash-Algorithmen
 - Nachrichtenauthentifizierungscodes
 - Asymmetrische Kryptographie und Digitale Signaturen
 - Schlüsselaustausch-Algorithmen
- Von der Kryptographie zu Zertifikaten und Protokollen.
 - Zertifikate
 - TLS
- Was ist ein VPN?
- OpenVPN

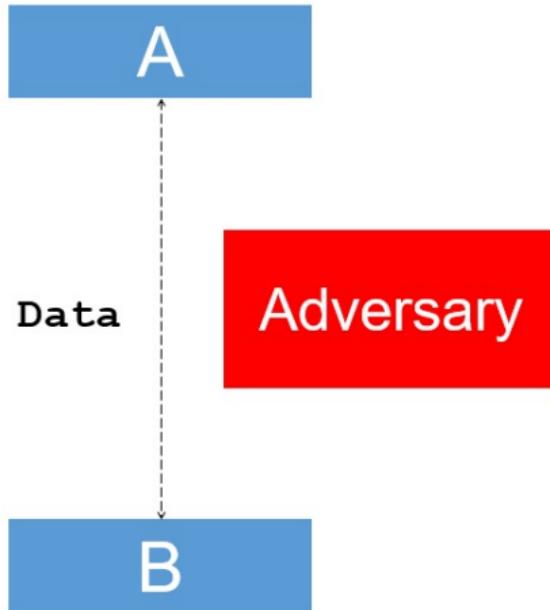


- Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin, Second Edition by Paul C. van Oorschot
- Handbook of Applied Cryptography, by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone
- Links:
 - <https://people.scs.carleton.ca/~paulv/toolsjewels.html>
 - <https://cacr.uwaterloo.ca/hac/>



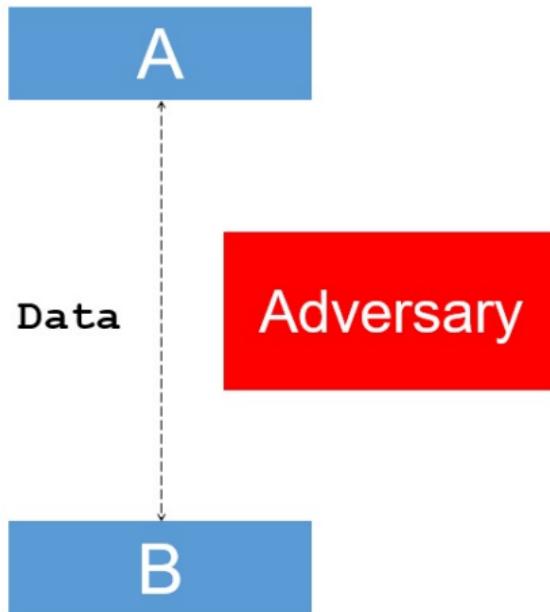
Was versuchen wir zu schützen?

- Ein Bedrohungsmodell beschreibt die Fähigkeiten, die ein Angreifer gegen eine Ressource einsetzen kann.
 - Damit können wir die Bedrohungen identifizieren, mit denen wir uns befassen müssen.
 - Weiters können wir bestimmte Bedrohungen ausdrücklich aus dem Betrachtungsbereich ausschließen.
- Das Bedrohungsmodell des Internets ist in RFC 3552 [15] dokumentiert.



Was versuchen wir zu schützen? (2)

- Angreiferinnen und Angreifer haben nahezu vollständige Kontrolle über den Kommunikationskanal.
- Typischerweise werden auch die Fähigkeiten der Angreifenden berücksichtigt - zum Beispiel durch die Unterscheidung zwischen 'On-Path'- und 'Off-Path'-Angriffen.
- Die Endsysteme, die am Protokollaustausch beteiligt sind, gelten als nicht kompromittiert.



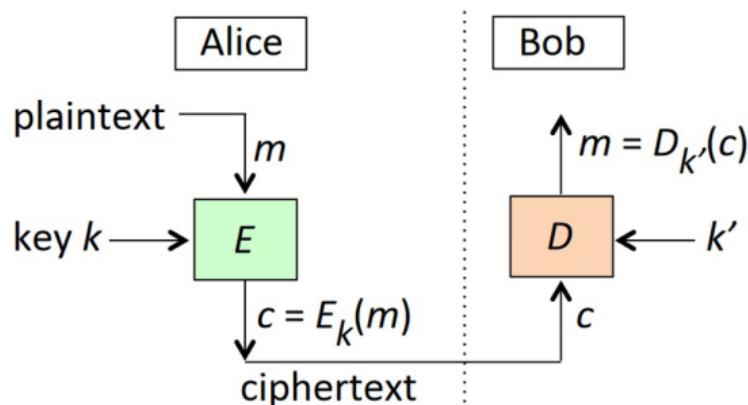
- Obwohl die Sicherheitsanforderungen jedes Systems einzigartig sind, treten bestimmte Anforderungen in vielen Protokollen auf.
- Um Lösungen zur Absicherung des Kommunikationskanals zwischen A und B bereitzustellen, werden häufig folgende Anforderungen gestellt:
 - Vertraulichkeit (Confidentiality)
 - Integrität (Integrity)
 - Verfügbarkeit (Availability) – RFC 3552 spricht von Authentifizierung (Authentication)
- Die Abkürzung CIA wird oft als Merkwort benutzt.
- TLS bietet eine Lösung zur Absicherung des Kommunikationskanals zwischen zwei Parteien, die diese (und noch weitere) Anforderungen abdeckt.



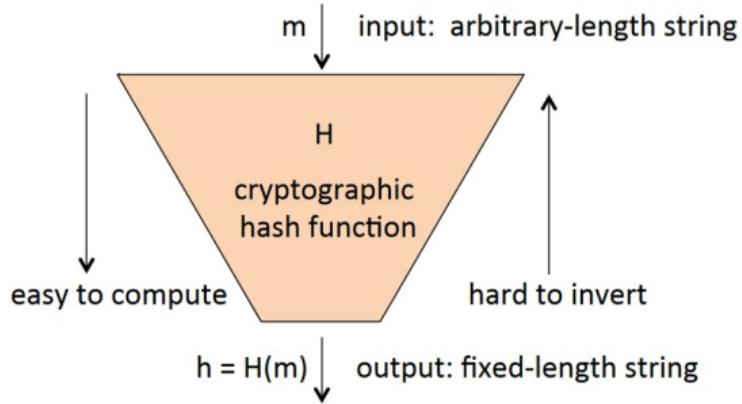
Eric Recorla, Autor von RFC 3552, TLS und vielen anderen Protokollen.

Bausteine

Verschlüsselung und Entschlüsselung

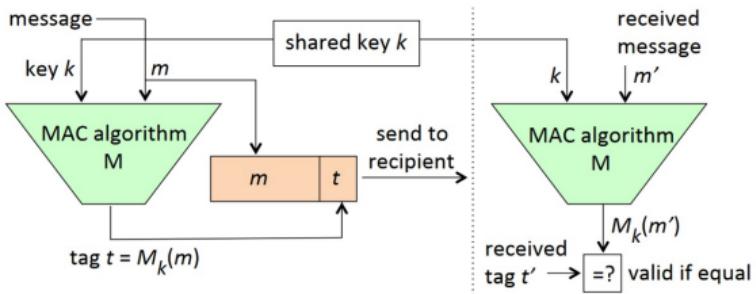


- Alice verbirgt den Inhalt der Nachricht mithilfe von Verschlüsselung (encryption), was einen Chiffretext (ciphertext) ergibt: $Enc_K(M) = C$
- Dazu benötigt Sie einen symmetrischen Schlüssel (key)
- Bob muss die Nachricht entschlüsseln (decryption): $Dec_K(C) = M$
- Eve schafft es die Nachricht abzufangen, erhält aber nur den Chiffretext
- Beispiel: Advanced Encryption Standard (AES) [10] - Galois/Counter Mode (GCM) [11]



- Kryptografische Hashfunktionen sind schlüssellose Verfahren.
- Gegeben ein Hashwert h , sollte es praktisch unmöglich sein, eine Nachricht m zu finden, so dass $H(m) = h$. (Preimage-Resistenz)
- Beispiel: Secure Hash Algorithm (SHA-256)
- Hashfunktionen finden Anwendung in digitalen Signaturen, bei der Erzeugung kryptografischer Zufallszahlen sowie in Key Derivation Functions (KDFs), bei denen ein geheimer Schlüssel als Eingabe genutzt wird.
- Beispiel: HMAC-based Extract-and-Expand Key Derivation Function (HKDF) [6]

Message Authentication Codes (MACs)



- MACs (Message Authentication Codes) gewährleisten die Datenintegrität, d.h. Veränderungen werden erkannt.
- Es handelt sich um ein Verfahren, das symmetrische Schlüssel verwendet.
- Beispiel: Hash-based Message Authentication Code (HMAC) mit SHA-256 [5].

Public Key Cryptography

644

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

Abstract—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

1976 [2]

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D , such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D . Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enci-

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R.L. Rivest, A. Shamir, and L. Adleman*

Abstract

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

1. Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
2. A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems.

A message is encrypted by representing it as a number M , raising M to a publicly specified power e , and then taking the remainder when the result is divided by the publicly specified product, n , of two large secret prime numbers p and q . Decryption is similar; only a different, secret, power d is used, where $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n .

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

1977 [17]



Öffentliche Parameter

p große Primzahl, g Generator von \mathbb{Z}_p^*

Alice

$x_A \leftarrow$ zufällig in $[1, p - 2]$

$$X = g^{x_A} \bmod p$$

sendet X an Bob.

Bob

$x_B \leftarrow$ zufällig in $[1, p - 2]$

$$Y = g^{x_B} \bmod p$$

sendet Y an Alice.

Gemeinsamer Schlüssel

$$K_A = Y^{x_A} \bmod p = g^{x_A x_B} \bmod p = X^{x_B} \bmod p = K_B$$

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

Alice: $x = 6$ Bob: $y = 15$

Öffentliche Werte berechnen:

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

Alice: $x = 6$ Bob: $y = 15$

Öffentliche Werte berechnen:

$$X = g^x \bmod p = 5^6 \bmod 23 = 8$$

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

$$\text{Alice: } x = 6 \quad \text{Bob: } y = 15$$

Öffentliche Werte berechnen:

$$X = g^x \bmod p = 5^6 \bmod 23 = 8$$

$$Y = g^y \bmod p = 5^{15} \bmod 23 = 19$$

Austausch:

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

$$\text{Alice: } x = 6 \quad \text{Bob: } y = 15$$

Öffentliche Werte berechnen:

$$X = g^x \bmod p = 5^6 \bmod 23 = 8$$

$$Y = g^y \bmod p = 5^{15} \bmod 23 = 19$$

Austausch:

Alice sendet $X = 8$, Bob sendet $Y = 19$

Alice ($x = 6$)

Bob ($y = 15$)

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

$$\text{Alice: } x = 6 \quad \text{Bob: } y = 15$$

Öffentliche Werte berechnen:

$$X = g^x \bmod p = 5^6 \bmod 23 = 8$$

$$Y = g^y \bmod p = 5^{15} \bmod 23 = 19$$

Austausch:

Alice sendet $X = 8$, Bob sendet $Y = 19$

$$\text{Alice (}x = 6\text{)} \xrightarrow{X = g^x \bmod p = 8} \text{Bob (}y = 15\text{)}$$

Beispiel: Diffie–Hellman mit kleinen Zahlen (1/2)

Gemeinsame öffentliche Parameter:

$$p = 23, \quad g = 5$$

Geheime Zufallszahlen:

$$\text{Alice: } x = 6 \quad \text{Bob: } y = 15$$

Öffentliche Werte berechnen:

$$X = g^x \bmod p = 5^6 \bmod 23 = 8$$

$$Y = g^y \bmod p = 5^{15} \bmod 23 = 19$$

Austausch:

Alice sendet $X = 8$, Bob sendet $Y = 19$

$$\begin{array}{ccc} X = g^x \bmod p = 8 & & \\ \text{Alice } (x = 6) \xleftrightarrow{} \text{Bob } (y = 15) & & \\ Y = g^y \bmod p = 19 & & \end{array}$$

Beispiel: Diffie–Hellman mit kleinen Zahlen (2/2)

Berechnung des gemeinsamen Geheimnisses:

Beispiel: Diffie–Hellman mit kleinen Zahlen (2/2)

Berechnung des gemeinsamen Geheimnisses:

$$Z_C = Y^x \bmod p = 19^6 \bmod 23 = 2$$

Beispiel: Diffie–Hellman mit kleinen Zahlen (2/2)

Berechnung des gemeinsamen Geheimnisses:

$$Z_C = Y^x \bmod p = 19^6 \bmod 23 = 2$$

$$Z_S = X^y \bmod p = 8^{15} \bmod 23 = 2$$

Beide erhalten dasselbe Geheimnis: $Z_C = Z_S = 2$

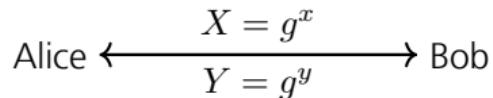
Beispiel: Diffie–Hellman mit kleinen Zahlen (2/2)

Berechnung des gemeinsamen Geheimnisses:

$$Z_C = Y^x \bmod p = 19^6 \bmod 23 = 2$$

$$Z_S = X^y \bmod p = 8^{15} \bmod 23 = 2$$

Beide erhalten dasselbe Geheimnis: $Z_C = Z_S = 2$



Erinnerung: $(g^y)^x = g^{y \cdot x} = g^{x \cdot y}$ — Exponenten werden multipliziert, nicht die Werte!

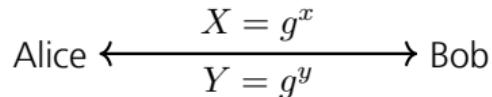
Beispiel: Diffie–Hellman mit kleinen Zahlen (2/2)

Berechnung des gemeinsamen Geheimnisses:

$$Z_C = Y^x \bmod p = 19^6 \bmod 23 = 2$$

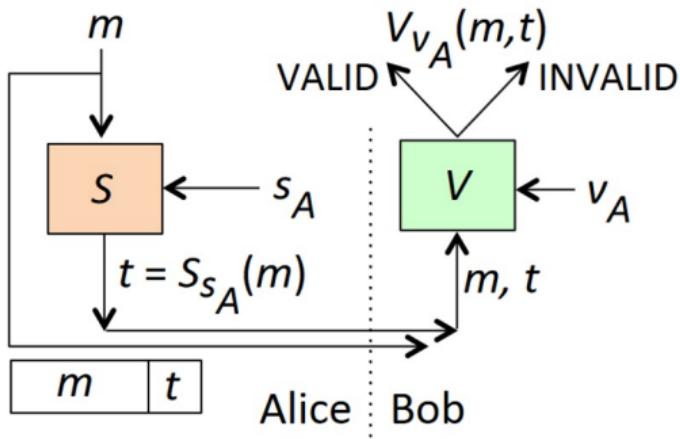
$$Z_S = X^y \bmod p = 8^{15} \bmod 23 = 2$$

Beide erhalten dasselbe Geheimnis: $Z_C = Z_S = 2$



$$Z_C = Y^x = g^{xy} \xrightarrow[\text{gleiches Ergebnis}]{} Z_S = X^y = g^{xy}$$

Erinnerung: $(g^y)^x = g^{y \cdot x} = g^{xy}$ — Exponenten werden multipliziert, nicht die Werte!



- Das asymmetrische Äquivalent zum MAC.
- Bietet auch Nichtabstreitbarkeit (Non-Repudiation).

s_A : signing private key (of Alice)

v_A : verification public key (of Alice)

Symmetrische Algorithmen

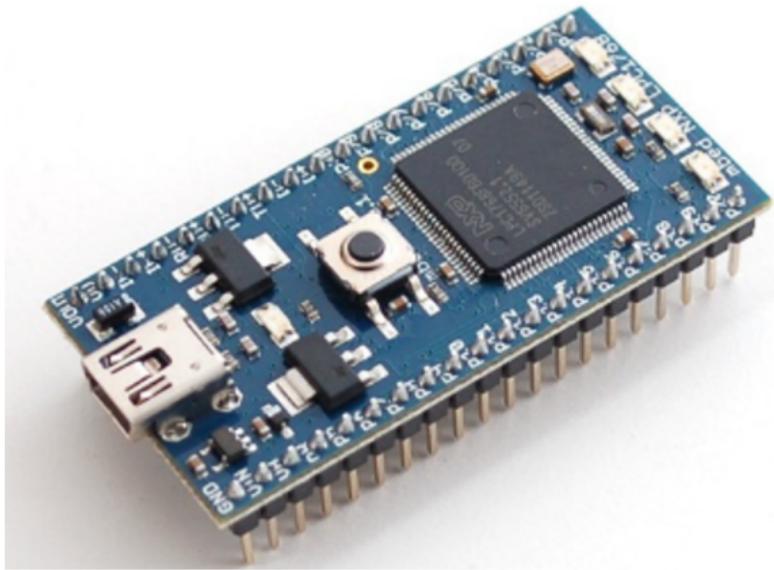
- Identischer Schlüssel für Verschlüsselung und Entschlüsselung
- $Enc_K(M) = C$ und $Dec_K(C) = M$
- Jahrtausende alt
- Typische Schlüssellänge: 128 Bit
- Effizient in der Rechenleistung und niedrige RAM-Anforderungen, geringer Overhead in der Nachrichtenlänge und kleine Codegröße.

Asymmetrische Algorithmen

- Verschiedene Schlüssel für Verschlüsselung und Entschlüsselung
- $Enc_K(M) = C$ und $Dec_{\bar{K}}(C) = M$
- 1976 (W. Diffie und M. Hellman)
- Typische Schlüssellänge: 3072 Bit (RSA)
- Gegenteil von symmetrischen Verfahren



Beispiel: Mikrocontroller



NXP LPC1768: Arm Cortex-M3 CPU, 96MHz mit
512 KB Flash und 32KB RAM

- SHA 256 (1024 Byte Nachricht): 0.6 msec
- AES-CBC-128 (1024 Byte Nachricht): 0.7 msec
- Signaturüberprüfung (P256r1, performance optimized): 458 msec
- Für Details siehe [19]

Von der Kryptographie zu Zertifikaten und Protokollen.

RFC 5280: Die X.509 Public Key Infrastruktur

- RFC 5280 [1] definiert die wesentlichen Zertifikatsformate, Zertifikatsperllisten und PKI-Konzepte.
- Die Spezifikation wurde in der IETF Arbeitsgruppe "Public-Key Infrastructure (X.509)"(pkix) entwickelt.
- Die Standardisierungsarbeiten werden nun in der IETF Arbeitsgruppe "Limited Additional Mechanisms for PKIX and SMIME"(lamps) fortgesetzt, siehe
<https://datatracker.ietf.org/wg/lamps/>.



Russ Housley, Mitautor von RFC 5280 sowie einer Vielzahl weiterer RFCs, ehemaliger IETF Chair, IETF Security Area Director und IAB Chair.

Das Public Key Zertifikat

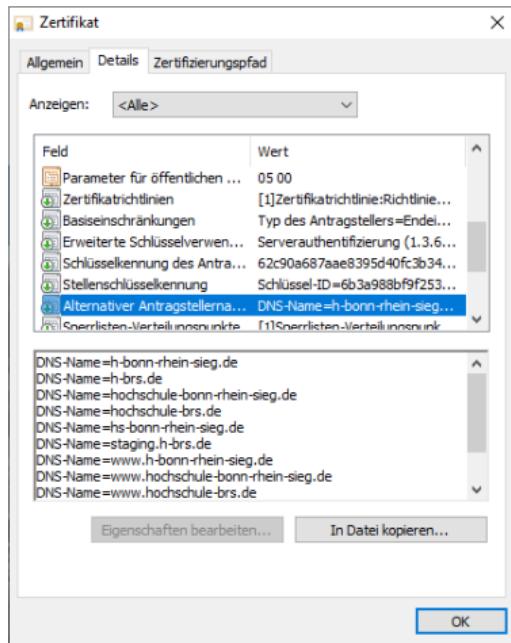
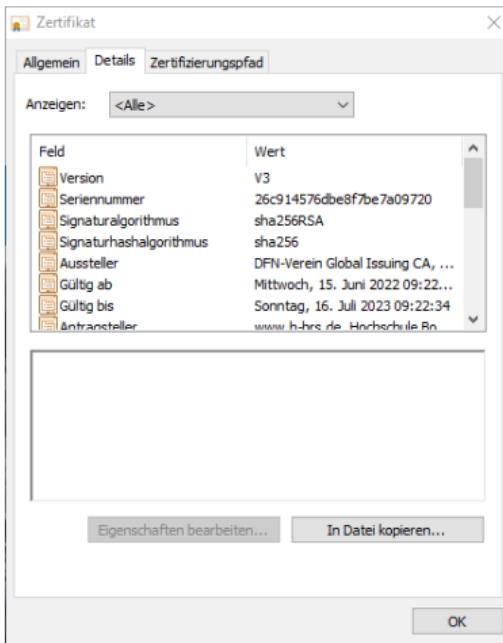
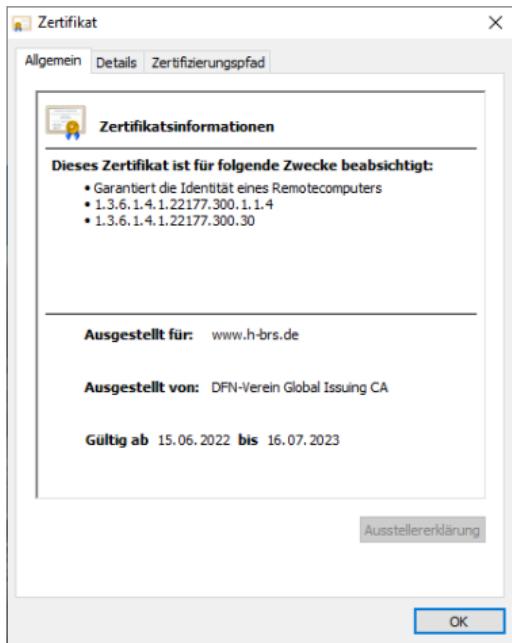
Field name	Contents or description
Version	X.509v3 or other versions
Serial-Number	uniquely identifies certificate, e.g., for revocation
Issuer	issuing CA's name
Validity-Period	specifies dates (Not-Before, Not-After)
Subject	owner's name
Public-Key info	specifies (Public-Key-Algorithm, Key-Value)
extension fields (optional)	Subject-Alternate-Name/SAN-list, Basic-Constraints, Key-Usage, CRL-Distribution-Points (and others)
Signature-Algorithm	(algorithmID, parameters)
Digital-Signature	signature of Issuer

Table 8.1: X.509v3 public-key certificate fields.

- Ein digitales Zertifikat bindet öffentlichen Schlüssel mit einem Subjekt.
- Ein Subjekt kann, zum Beispiel über einen Domainnamen identifiziert werden. Typischerweise werden noch weitere Attribute inkludiert.
- Durch die digitale Signatur einer vertrauenswürdigen Partei, dem Herausgeber des Zertifikats, wird der Inhalt gegen Modifikation geschützt.

Abbildung aus Kapitel 8 aus dem Buch "Tools & Jewels" von Paul van Oorschot [21].

Beispiele von digitalen Zertifikaten

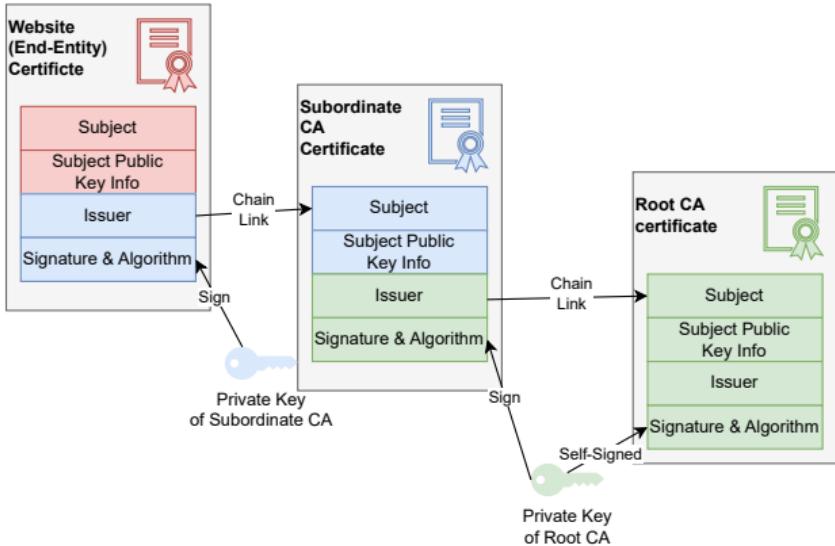


- **End Entity:** Das Subjekt, das ein Zertifikat benötigt (z. B. Server, Nutzer, IoT Gerät).
- **Registration Authority (RA):** Prüft die Identität des End Entity sowie die zugehörigen Attribute.
- **Certification Authority (CA):**
 - signiert nach erfolgreicher Prüfung durch die RA die Zertifikatsanfrage,
 - bestätigt dadurch die Bindung zwischen öffentlichem Schlüssel und Identität,
 - erstellt das Zertifikat durch Signatur mit ihrem privaten Schlüssel.
- **Public Key Infrastructure (PKI):** Stellt weitere Dienste bereit, z. B.
 - Sperrlisten (CRLs),
 - Online-Statusabfragen (OCSP),
 - Schlüssel- und Zertifikatsmanagement.

Zentrale Fragestellungen:

- Muss ich die Zertifikate jeder Website kennen?
- Wie bekomme ich das aktuelle Zertifikat einer Website?
- Muss ich den öffentlichen Schlüssel jeder CA kennen?
- Wie stelle ich sicher, dass ein Zertifikat noch gültig ist?
- Wie funktioniert Vertrauen zwischen verschiedenen CAs?

Vertrauenskette und Path Validation



- Das CA/Browser-Vertrauensmodell erfordert nur das Vertrauen in die Root-CA-Zertifikate.
- Diese Root-CA-Zertifikate werden als „Trust Anchors“ bezeichnet.
- Die Verteilung dieser Trust Anchors erfolgt typischerweise durch die Browsersoftware.
- Alle anderen Zertifikate werden im Rahmen des TLS-Handshakes ausgetauscht und durch den Path Validation Algorithmus verifiziert.

Public Key Cryptography Standards (PKCS)

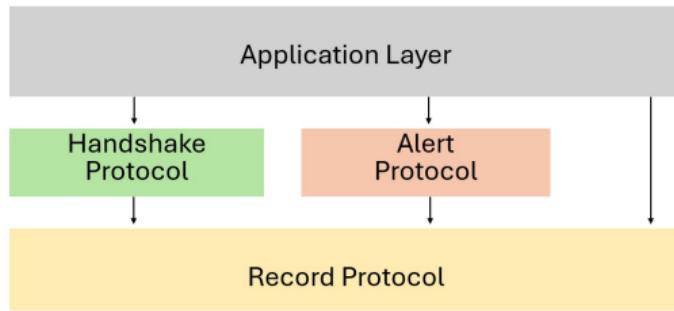
PKCS	Beschreibung	RFC
PKCS #1	RSA Cryptographic Standard	RFC 8017 [7]
PKCS #5	Password-based Cryptography	RFC 8018 [8]
PKCS #7	Cryptographic Message Syntax (CMS) used to digitally sign, digest, authenticate, and encrypt arbitrary content	RFC 5652 [4])
PKCS #8	Private-Key Information Syntax Standard	RFC 5958 [20]
PKCS #9	Selected Attribute Types	RFC 2985 [13]
PKCS #10	Certification Request Standard	RFC 2986 [12]
PKCS #11	Cryptographic Token Interface	Standardisiert in OA-SIS [3]
PKCS #12	Personal Information Exchange Syntax Standard	RFC 7292 [9]

Loren Kohnfelder beschrieb ursprünglich das Konzept der Zertifikate und der PKI in seiner Dissertation am MIT. Burt Kaliski verfasste die ersten Spezifikationen, als er bei RSA Inc. war.

- TLS ist ein flexibles Protokoll mit vielen Optionen, um ein breites Einsatzgebiet abzudecken.
- TLS existiert in verschiedenen Versionen; die aktuelle Version ist 1.3. TLS (und sein Vorgänger SSL) wird seit über 30 Jahren entwickelt.
- Der Austausch erfolgt in zwei Phasen:
 - **Phase 1:** Komplexer Handshake mit mehreren Nachrichten, bei dem typischerweise asymmetrische Kryptografie zum Einsatz kommt. Ergebnis: symmetrische Schlüssel zur Verwendung mit den ausgehandelten Algorithmen.
 - **Phase 2:** Symmetrische Kryptografie zum Schutz des Datenaustauschs - schnell und mit geringem Overhead.
- TLS bietet eine sichere Aushandlung der zu verwendeten kryptografischen Algorithmen und von Erweiterungen.



TLS Record Protokoll



- Alle Nachrichten werden über das **TLS Record Protocol** verschickt.
- Falls symmetrische Schlüssel vorhanden sind, werden die Daten verschlüsselt. Applikationsdaten werden nicht unverschlüsselt verschickt - manche Handshake- und Alert-Nachrichten schon.
- Das Record Protocol bietet:
 - Vertraulichkeit
 - Integrität
 - Schutz der Reihenfolge (Replay Protection)
 - Längenverschleierung (Traffic Flow Confidentiality / Length concealment)

TLS Handshake Protokoll

Client

Server

```
clientHello (+key_share +signature_algorithms,  
+supported_groups, +supported_versions)
```

```
serverHello (+key_share),  
{EncryptedExtensions (+supported_groups*)},  
{CertificateRequest*}, {Certificate*},  
{CertificateVerify*}, {Finished},  
[Application Data*]
```

```
{Certificate*}, {CertificateVerify*},  
{Finished}, [Application Data]
```

```
[Application Data]
```

Legend: *: optional message, []: Not a handshake message, {}: Encrypted message

- In der Abbildung ist die beliebteste Variante dargestellt, die zur Authentifizierung Public-Key-Zertifikate verwendet. Dieser Austausch nutzt verpflichtend den Diffie-Hellman-Algorithmus zum Schlüsselaustausch.

- Server-seitige Authentifizierung ist verpflichtend; die client-seitige Authentifizierung ist jedoch optional.

- Nach erfolgreichem Abschluss des TLS-Handshakes stehen beiden Parteien symmetrische Schlüssel zur Verfügung, mit denen die Applikationsdaten geschützt werden können.

Wireshark Trace

*WLAN

Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telefonie Wireless Tools Hilfe

ip.addr == 193.99.144.85

No.	Time	Source	Destination	Protocol	Length	Info
93	1.209210	172.16.55.243	193.99.144.85	TCP	66	57593 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
95	1.214881	193.99.144.85	172.16.55.243	TCP	66	443 → 57593 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 WS=4 SACK_PERM
96	1.215088	172.16.55.243	193.99.144.85	TCP	54	57593 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
97	1.217187	172.16.55.243	193.99.144.85	TLSv1.3	714	Client Hello (SNI=www.heise.de)
98	1.227972	193.99.144.85	172.16.55.243	TCP	56	443 → 57593 [ACK] Seq=1 Ack=661 Win=5040 Len=0
99	1.227972	193.99.144.85	172.16.55.243	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
100	1.227972	193.99.144.85	172.16.55.243	TLSv1.3	1514	Application Data
101	1.227972	193.99.144.85	172.16.55.243	TLSv1.3	282	Application Data, Application Data
102	1.228091	172.16.55.243	193.99.144.85	TCP	54	57593 → 443 [ACK] Seq=661 Ack=3149 Win=131328 Len=0
104	1.242395	172.16.55.243	193.99.144.85	TLSv1.3	118	Change Cipher Spec, Application Data
105	1.243060	172.16.55.243	193.99.144.85	TLSv1.3	224	Application Data
106	1.243127	172.16.55.243	193.99.144.85	TLSv1.3	533	Application Data
107	1.247255	193.99.144.85	172.16.55.243	TCP	56	443 → 57593 [ACK] Seq=3149 Ack=725 Win=5104 Len=0
108	1.247490	193.99.144.85	172.16.55.243	TLSv1.3	103	Application Data
109	1.247648	172.16.55.243	193.99.144.85	TLSv1.3	85	Application Data
110	1.247757	193.99.144.85	172.16.55.243	TCP	56	443 → 57593 [ACK] Seq=3198 Ack=895 Win=5272 Len=0
111	1.248077	193.99.144.85	172.16.55.243	TCP	56	443 → 57593 [ACK] Seq=3198 Ack=1374 Win=5752 Len=0
112	1.249127	193.99.144.85	172.16.55.243	TLSv1.3	1514	Application Data, Application Data
113	1.249166	172.16.55.243	193.99.144.85	TCP	54	57593 → 443 [ACK] Seq=1405 Ack=4658 Win=131328 Len=0
114	1.253739	193.99.144.85	172.16.55.243	TLSv1.3	1514	Application Data
115	1.253739	193.99.144.85	172.16.55.243	TLSv1.3	1514	Application Data

Trace ohne Verwendung von SSLKEYLOGFILE [18].

Referenzen

Internet Engineering Task Force (IETF)
Request for Comments: 8446
Obsoletes: 5077, 5246, 6961
Updates: 5705, 6066
Category: Standards Track
ISSN: 2070-1721

The Transport Layer Security (TLS) Protocol Version 1.3

Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

This document updates RFCs 5705 and 6066, and obsoletes RFCs 5077, 5246, and 6961. This document also specifies new requirements for TLS 1.2 implementations.

RFC 8446 [14]

- Die Standardisierung von TLS und seinen Erweiterungen erfolgte und erfolgt weiterhin in der IETF TLS-Arbeitsgruppe (siehe <https://datatracker.ietf.org/wg/tls/about/>).

E. Rescorla
Mozilla
August 2018

Internet Engineering Task Force (IETF)
Request for Comments: 9147
Obsoletes: 6347
Category: Standards Track
ISSN: 2070-1721

E. Rescorla
Mozilla
H. Tschofenig
Arim Limited
N. Modadugu
Google, Inc.
April 2022

The Datagram Transport Layer Security (DTLS) Protocol Version 1.3

Abstract

This document specifies version 1.3 of the Datagram Transport Layer Security (DTLS) protocol. DTLS 1.3 allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

The DTLS 1.3 protocol is based on the Transport Layer Security (TLS) 1.3 protocol and provides equivalent security guarantees with the exception of order protection / non-replayability. Datagram semantics of the underlying transport are preserved by the DTLS protocol.

This document obsoletes RFC 6347.

RFC 9147 [16]

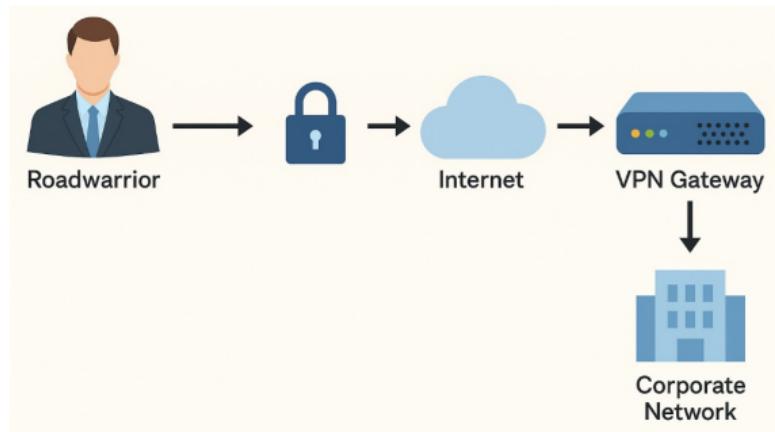
Virtual Private Network (VPN)

- **Sicherheitsziele:** Authentifizierung, Vertraulichkeit, Integrität, Replay-Schutz
- Populäre Einsatzszenarien:
 - **Road-Warrior** (z.B. Mitarbeiterin auf Dienstreise verbinden sich mit dem Firmennetzwerk)
 - Host-to-Host (z.B. Absicherung der Kommunikation zwischen Cloud-Servern)
 - Site-to-Site (z.B. Netzwerke verschiedener Firmenstandorte miteinander verbinden)
- VPNs erfreuen sich zur Zeit großer Beliebtheit und werden oft auf Social Media beworben. Die beworbenen Sicherheitseigenschaften von VPN-Produkten werden oft übertrieben dargestellt.
- **Gängige Protokolle:**
 - IPsec (IP Security)
 - **OpenVPN**
 - WireGuard
 - Tor (benutzt mehrere Tunnels verschachtelt ineinander)
 - Traversal Using Relays around NAT (TURN) (für Echtzeitapplikationen wie VoIP)
 - Multiplexed Application Substrate over QUIC Encryption (MASQUE)

- **Grundprinzip:** Aufbau eines verschlüsselten 'Tunnels' zwischen Client und VPN-Gateway

- **Typische Einsatzszenarien:**

- Remote-Arbeit (z.B. Zugriff auf das Firmennetzwerk)
- Absicherung der Kommunikation in unsicheren öffentlichen Netzwerken (z.B. WLAN-Hotspots in Cafés)
- Umgehung von Geoblocking: Viele Webseitenbetreiber prüfen den Standort des Endgeräts anhand der IP-Adresse, um den Zugriff auf urheberrechtlich geschützte Inhalte zu kontrollieren.

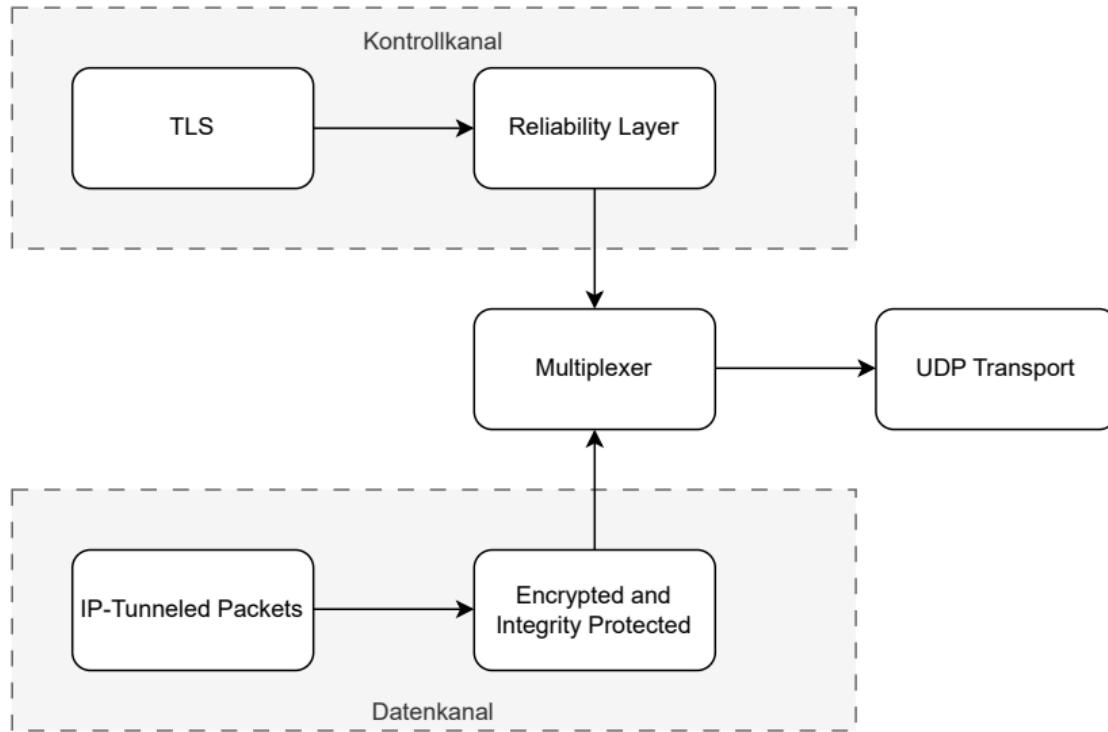


OpenVPN: Übersicht

- OpenVPN ist eine Implementierung für Roadwarrior-VPN-Szenarien.
- Es wurde erstmals im Jahr 2001 veröffentlicht und wird seither aktiv weiterentwickelt. OpenVPN wird auch an der Hochschule als VPN-Lösung eingesetzt.
- Zur Authentifizierung mit dem VPN-Gateway und zum Aufbau eines sicheren Kommunikationskanals wird TLS verwendet (als Kontrollkanal).
- Anstatt Applikationsdaten direkt über den TLS Record Layer zu übertragen, nutzt OpenVPN ein eigenes, UDP-basiertes Protokoll (als Datenkanal).
- Der über TLS aufgebaute sichere Kanal dient zur Verteilung symmetrischer Schlüssel.
- Der Ansatz, TLS für Authentifizierung und Schlüsselmanagement zu nutzen, um andere Protokolle abzusichern, wurde später in weiteren Bereichen übernommen - etwa bei Wi-Fi (Enterprise) und in der VoIP-Sicherheit.
- Die Möglichkeit, TCP zu verwenden, wurde später ergänzt, um Firewalls zu umgehen. OpenVPN kann beispielsweise so konfiguriert werden, dass es über Port 443 kommuniziert.



OpenVPN: Architektur



OpenVPN: Architektur / Zusammenfassung

- OpenVPN unterscheidet zwischen einem Kontrollkanal und einem Datenkanal.

■ **Kontrollkanal:**

- Nutzung von TLS zur gegenseitigen Authentisierung
- Aushandlung kryptographischer Parameter
- Optionaler Schutz der Kontrollpakete:
 - --tls-auth: fügt einen vorgelagerten HMAC als Paketfilter hinzu
 - --tls-crypt: verschlüsselt zusätzlich den gesamten Kontrollkanal (inkl. HMAC)
 - dient als DoS-/Portscan-Schutz
 - gehört nicht zum TLS Record Layer

■ **Datenkanal:**

- Transport vollständiger IP-Pakete über das TUN-Interface
- Symmetrische Verschlüsselung und Integritätsschutz über OpenSSL EVP

■ **Multiplexing:**

- Kontroll- und Datenpakete werden zu einem gemeinsamen Strom zusammengeführt
- Transport über UDP (oder optional TCP)
- Multiplexer trennt beim Empfang die Pakete wieder nach Kanal auf

Akronyme I

- AES** Advanced Encryption Standard
- CBC** Cipher Block Chaining
- CMS** Cryptographic Message Syntax
- CRL** Certificate Revocation List
- GCM** Galois/Counter Mode
- HKDF** HMAC-based Extract-and-Expand Key Derivation Function
- HMAC** Hash-based Message Authentication Code
- KDF** Key Derivation Function
- OCSP** Online Certificate Status Protocol
- PKCS** Public Key Cryptography Standards
- PKI** Public-Key-Infrastruktur
- RA** Registration Authority
- RSA** Rivest-Shamir-Adleman
- SHA** Secure Hash Algorithm
- TLS** Transport Layer Security
- VPN** Virtual Private Network
- X.509** ITU-T Recommendation X.509

Quellen I

- [1] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008.
<http://www.rfc-editor.org/rfc/rfc5280.txt>.
- [2] Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Transactions on Information Theory* 22, 6 (1976), 644–654.
- [3] Gleeson, S., Zimman, C., Griffin, R., and Hudson, T. Pkcs #11 cryptographic token interface base specification version 2.40 plus errata 01. Latest version: <http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/pkcs11-base-v2.40.html>, May 2016. OASIS Standard Incorporating Approved Errata 01.
- [4] Housley, R. Cryptographic Message Syntax (CMS). RFC 5652, September 2009.
- [5] Krawczyk, H., Bellare, M., and Canetti, R. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, February 1997.
- [6] Krawczyk, H., and Eronen, P. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, May 2010.
- [7] Moriarty, K., Kaliski, B., Jonsson, J., and Rusch, A. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, November 2016.
- [8] Moriarty, K., Kaliski, B., and Rusch, A. PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018, January 2017.

Quellen II

- [9] Moriarty, K., Nyström, M., Parkinson, S., Rusch, A., and Scott, M. PKCS #12: Personal Information Exchange Syntax v1.1. RFC 7292, July 2014.
- [10] National Institute of Standards and Technology. Advanced Encryption Standard. *NIST FIPS PUB 197* (November 2001).
- [11] National Institute of Standards and Technology. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. *NIST SP 800-38D* (November 2007).
- [12] Nyström, M., and Kaliski, B. PKCS #10: Certification Request Syntax Specification Version 1.7. RFC 2986, November 2000.
- [13] Nyström, M., and Kaliski, B. PKCS #9: Selected Object Classes and Attribute Types Version 2.0. RFC 2985, November 2000.
- [14] Rescorla, E. The transport layer security (tls) protocol version 1.3. RFC 8446, RFC Editor, August 2018.
- [15] Rescorla, E., and Korver, B. Guidelines for Writing RFC Text on Security Considerations. RFC 3552, July 2003.
- [16] Rescorla, E., Tschofenig, H., and Modadugu, N. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147, April 2022.
- [17] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (February 1978), 120–126.



Quellen III

- [18] Thomson, M., Rosomakho, Y., and Tschofenig, H.
The SSLKEYLOGFILE Format for TLS.
Internet-Draft draft-ietf-tls-keylogfile-05, Internet Engineering Task Force, June 2025.
Work in Progress.
- [19] Tschofenig, T., and Pegourie-Gonnard, M.
Performance of state-of-the-art cryptography on arm-based microprocessors.
NIST 2015 workshop on Lightweight Cryptography (July 2015).
- [20] Turner, S.
Asymmetric Key Packages.
RFC 5958, August 2010.
- [21] Van Oorschot, P. C.
Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin.
Information Security and Cryptography. Springer International Publishing, Cham, 2021.