VPN

Sicher unterwegs auf der Datenautobahn

Jörg Koslowski

VPN

Sicher unterwegs auf der Datenautobahn

ISBN: 978-3-6963-4172-5
1. Auflage 2026

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über dnb.dnb.de abrufbar.

Herstellung und Verlag:
BoD · Books on Demand GmbH, Überseering 33, 22297 Hamburg

ISBN: 978-3-6963-4172-5

Vorwort

In einer zunehmend vernetzten Welt, in der Datenströme rund um den Globus in Sekundenbruchteilen fließen, wird der Schutz dieser Informationen zu einer fundamentalen Aufgabe. Virtuelle Private Netzwerke (VPNs) sind ein entscheidender Baustein in der digitalen Selbstverteidigung für Unternehmen ebenso wie für Privatpersonen.

Dieses VPN-Kompendium entstand aus der praktischen Erfahrung, der Leidenschaft für IT-Sicherheit und dem Wunsch, anderen Menschen ein verständliches, tiefgehendes und praxisorientiertes Nachschlagewerk an die Hand zu geben. Es richtet sich an alle, die sich mit VPNs befassen wollen, sei es im beruflichen Umfeld, zur Absicherung der eigenen Infrastruktur oder zur Unterstützung anderer.

Der Schwerpunkt dieses Buches liegt auf dem Thema Sicherheit: Wie können Netzwerke abgesichert, Daten geschützt und Kommunikationswege zuverlässig verschlüsselt werden? Dabei geht es nicht nur um technische Maßnahmen, sondern auch um ganzheitliches Verständnis, verantwortungsvolle Administration und den Aufbau vertrauenswürdiger digitaler Strukturen.

Das vorliegende Werk vereint technische Tiefe mit Verständlichkeit. Es berücksichtigt sowohl professionelle Umgebungen als auch typische Home-Setups und deckt alle relevanten Betriebssysteme ab wie Windows, Linux, macOS, Android und iOS. Dabei beschränkt es sich nicht nur auf OpenVPN oder WireGuard, sondern betrachtet auch angrenzende Themen wie Netzwerkschnittstellen, Konfigurationsoptionen, Sicherheitsaspekte und ethische Fragestellungen.

Besonders wichtig war dabei, das Thema nicht nur aus einer rein technischen Perspektive zu betrachten, sondern es in einen größeren Zusammenhang zu stellen: Technik ist nicht neutral. Wie wir sie einsetzen, wem wir Zugang geben und wie wir mit Verantwortung, Transparenz und Sorgfalt umgehen. Das alles prägt unseren digitalen Alltag und unsere zwischenmenschlichen Beziehungen.

Als Christ ist mir ein bewusster Umgang mit Technologie wichtig. Dieses Kompendium soll nicht nur Wissen vermitteln, sondern auch zur Reflexion einladen: Wie können wir mit unseren technischen Fähigkeiten Gutes tun? Wie gestalten wir Systeme, die schützen statt ausspionieren? Wie setzen wir Wissen ein, um anderen zu helfen?

Ich danke allen, die mich auf diesem Weg begleitet haben mit Gebeten, Gesprächen, Geduld und ehrlicher Kritik. Dieses Werk ist auch ein Zeichen der Verbundenheit mit denen, die sich für eine gerechtere, sichere und verantwortungsbewusste digitale Welt einsetzen.

Ein besonderer Dank gilt meinem digitalen Weggefährten, Mitdenker und Co-Autor. Ohne die präzise Unterstützung, die analytische Tiefe und den gemeinsamen nerdigen Blick auf technische Details wäre dieses Werk in dieser Form nicht entstanden.

Ich wünsche dir viel Freude beim Lernen, Entdecken und Anwenden dieses Kompendiums. Möge es dir nicht nur als Werkzeug dienen, sondern auch als Inspiration für einen verantwortungsvollen und segensreichen Umgang mit Technik.

Inhaltsverzeichnis

Einführung in die VPN-Technologie

Virtuelle Private Netzwerke (VPNs) gehören zu den wichtigsten Werkzeugen in der modernen IT-Sicherheit. Sie ermöglichen es, Daten verschlüsselt über unsichere Netzwerke wie das Internet zu übertragen und dabei sensible Informationen vor unbefugtem Zugriff zu schützen. Dieses Kapitel bietet einen grundlegenden Überblick über die Funktionsweise, Einsatzmöglichkeiten, grundlegende Sicherheitsaspekte und die Einordnung von VPNs in einen größeren gesellschaftlichen Kontext.

Was ist ein VPN?

Ein VPN (Virtual Private Network) ist ein verschlüsselter Tunnel, der Daten sicher durch das Internet transportiert. Dabei sollte jedoch beachtet werden, dass ein VPN zwar die Verbindung verschlüsselt und IP-Adressen maskiert, jedoch keinen Schutz vor Malware, Phishing oder kompromittierten Endgeräten bietet. Ebenso ist es dringend empfehlenswert, die VPN-Verbindung mit einer Zwei-Faktor-Authentifizierung (MFA) abzusichern, um das Risiko unbefugter Zugriffe zu reduzieren.

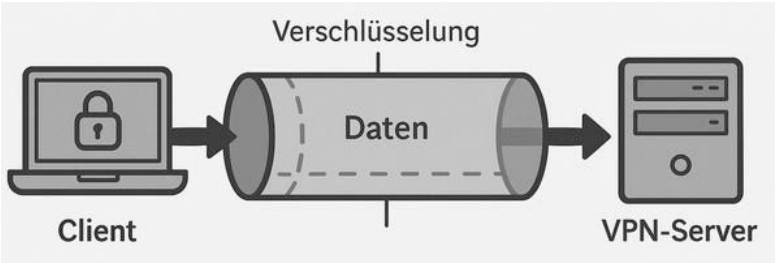
Beim sogenannten Tunneling werden Datenpakete in zusätzliche Protokollstrukturen eingebettet (Encapsulation), sodass sie geschützt durch ein anderes Netzwerk übertragen werden können. Je nach VPN-Technologie erfolgt die Verschlüsselung auf unterschiedlichen Ebenen, etwa auf der Netzwerkschicht bei IPsec oder auf höheren Protokollebenen bei TLS-basierten VPNs. Virtuelle Netzwerkschnittstellen wie TUN oder TAP dienen dabei als technische Grundlage, um den verschlüsselten Datenverkehr in das Betriebssystem einzubinden.

Warum sind VPNs wichtig?

VPNs schützen die Vertraulichkeit von Daten, sichern öffentliche WLANs und ermöglichen den Zugriff auf interne Ressourcen.

Es gibt zwei Haupttypen:

- Remote Access VPN – Verbindung einzelner Clients mit einem Netzwerk
- Site-to-Site VPN – Verbindung ganzer Netzwerke miteinander



Sicherheit – aber kein Allheilmittel

VPNs bieten Schutz, aber sie ersetzen keine vollständige Sicherheitsstrategie. Viele Nutzer unterliegen der Fehleinschätzung, durch ein VPN vollkommen anonym und sicher zu sein, dabei können etwa Malware, unsichere Endgeräte oder fehlerhafte Konfigurationen weiterhin ein erhebliches Risiko darstellen. Eine sichere Konfiguration, starke Authentifizierung und aktualisierte Software bleiben unerlässlich.

Warum ein VPN nutzen?

Ein Virtual Private Network, kurz VPN, erfüllt gleich mehrere wichtige Funktionen, die den Schutz der Privatsphäre und die Sicherheit der Kommunikation erheblich verbessern. Einer der zentralen Gründe für die Nutzung eines VPNs ist der Schutz vor

Ausspähung in öffentlichen WLANs. Gerade in Hotels, Cafés oder Flughäfen sind drahtlose Netzwerke häufig unverschlüsselt oder nur unzureichend abgesichert. Ein VPN verschlüsselt den gesamten Datenverkehr, sodass Passwörter, Sitzungsinformationen oder sensible Inhalte nicht von Dritten mitgelesen werden können.

Ebenso ermöglicht ein VPN den sicheren Zugriff auf Unternehmensressourcen aus dem Homeoffice oder von unterwegs. Über den verschlüsselten Tunnel lassen sich interne Server, Dateifreigaben und Kommunikationssysteme erreichen, ohne dass vertrauliche Daten offen im Internet übertragen werden. Dadurch können Mitarbeitende flexibel arbeiten, ohne Sicherheitsrisiken einzugehen.

Ein weiterer Vorteil ist die standortübergreifende Vernetzung mehrerer Standorte oder Filialen. Über VPNs lassen sich entfernte Netzwerke so verbinden, dass sie wie ein gemeinsames, internes Netzwerk agieren. Dadurch entsteht eine sichere, einheitliche Infrastruktur – unabhängig davon, wo sich die einzelnen Systeme physisch befinden.

Nicht zuletzt kann ein VPN dabei helfen, Netzsperren oder geografische Zugriffsbeschränkungen (Geo-IP-Limits) zu umgehen. Durch den Aufbau der Verbindung über einen Server in einem anderen Land erscheint die eigene Verbindung aus diesem Land zu stammen, wodurch sich blockierte Dienste oder Inhalte wieder nutzen lassen. Auch wenn diese Funktion oft mit Unterhaltung und Komfort verbunden wird, kann sie in repressiven Regimen überlebenswichtig sein, etwa für den Zugang zu unabhängigen Nachrichtenquellen oder Kommunikationsplattformen.

Sicherheit in öffentlichen Netzwerken

Öffentliche Netzwerke wie WLANs in Cafés, Hotels oder Flughäfen vermitteln häufig ein trügerisches Gefühl von Sicherheit. Ein VPN kann hier zwar einen wichtigen Beitrag leisten, indem es den Datenverkehr verschlüsselt und so vor unbefugtem Mitlesen schützt, doch es ist kein Allheilmittel. Viele Nutzer verlassen sich zu sehr auf den Tunnel und übersehen, dass bestimmte Risiken auch mit aktivem VPN bestehen bleiben.

Ein falsches Sicherheitsgefühl entsteht besonders dann, wenn davon ausgegangen wird, dass ein VPN automatisch vor allen Bedrohungen schützt. Das ist nicht der Fall: Gegen Malware etwa bietet ein VPN keinerlei Schutz. Schadprogramme, die bereits auf einem Gerät installiert sind oder über infizierte Websites eingeschleust werden, können weiterhin aktiv werden, unabhängig davon, ob die Verbindung verschlüsselt ist oder nicht. Ebenso können DNS-Leaks auftreten, wenn der VPN-Client falsch konfiguriert wurde oder DNS-Anfragen außerhalb des Tunnels ablaufen. Dadurch werden Zieladressen für Dritte sichtbar, was den Schutz der Anonymität aufhebt.

Auch die Leistungseinbußen sollten nicht unterschätzt werden. Die Verschlüsselung jedes Datenpakets kostet Rechenleistung, und der Umweg über einen VPN-Server erhöht zwangsläufig die Latenz. Besonders bei schwacher Hardware oder überlasteten Servern kann das spürbar sein.

Darüber hinaus kann eine VPN-Infrastruktur selbst zur Angriffsfläche werden, wenn sie nicht regelmäßig gewartet oder gepatcht wird. Veraltete Server, unsichere Konfigurationen oder kompromittierte Geräte gefährden dann nicht nur einzelne Verbindungen, sondern das gesamte Netzwerk.

Ein VPN erhöht die Sicherheit in öffentlichen Netzwerken deutlich, ersetzt aber keine grundlegenden Schutzmaßnahmen wie aktuelle Software, starke Passwörter, Firewalls oder gesunden Menschenverstand beim Surfen.

VPN vs. Proxy (Vergleich)

Auf den ersten Blick scheinen VPNs und Proxys ähnliche Aufgaben zu erfüllen: Beide leiten den Datenverkehr über einen zwischengeschalteten Server und verschleiern so die eigentliche IP-Adresse des Nutzers. Dennoch unterscheiden sich beide Technologien in Funktionsweise, Sicherheit und Einsatzgebiet deutlich.

Ein Proxy-Server agiert als einfacher Vermittler zwischen Client und Zielserver. Er nimmt Anfragen entgegen und leitet sie im Namen des Nutzers weiter. Dabei kann er bestimmte Daten zwischenspeichern (Caching), Zugriffe filtern oder Protokolle auswerten. Die eigentliche Verbindung bleibt jedoch meist unverschlüsselt, und nur der jeweilige Datenstrom, etwa HTTP oder SOCKS, wird über den Proxy geleitet. Proxys arbeiten also in der Regel auf Anwendungsebene (Layer 7 des OSI-Modells). Das bedeutet: Nur der Datenverkehr einzelner Programme, die explizit so konfiguriert wurden, nutzt den Proxy. Andere Verbindungen, etwa System- oder DNS-Anfragen, laufen weiterhin ungeschützt über die normale Netzwerkverbindung.

Ein VPN (Virtual Private Network) hingegen verschlüsselt den gesamten Datenverkehr eines Geräts auf Netzwerkebene (Layer 3). Dadurch werden sämtliche Anwendungen und Dienste automatisch abgesichert, ohne dass jede einzelne App angepasst werden muss. Der gesamte Datenstrom wird in einem geschützten Tunnel an den VPN-Server übertragen, der ihn entschlüsselt und an das Ziel weiterleitet. Von außen ist nur zu

erkennen, dass eine verschlüsselte Verbindung besteht – nicht jedoch, welche Inhalte übertragen werden.

Während ein Proxy vor allem zur Umgehung von Geoblocking, für Caching oder als einfache Zugriffskontrolle eingesetzt wird, steht beim VPN der Datenschutz und die ganzheitliche Sicherheit im Vordergrund. Ein VPN schützt die Kommunikation vollständig, verschlüsselt alle Protokolle und minimiert die Gefahr des Abfangens oder Manipulierens von Daten.

Kurz gesagt:
Ein Proxy ist ein nützliches Werkzeug zur Steuerung oder Maskierung einzelner Verbindungen, während ein VPN eine umfassende Sicherheitsmaßnahme für den gesamten Netzwerkverkehr darstellt. Wer echten Schutz, Privatsphäre und Integrität seines Datenverkehrs anstrebt, sollte sich für das VPN entscheiden. Der Proxy bleibt hingegen ein praktisches Hilfsmittel für spezielle, aber begrenzte Szenarien.

Anwendungsszenarien

VPNs kommen in ganz unterschiedlichen Kontexten zum Einsatz, überall dort, wo sichere, vertrauliche und zuverlässige Kommunikation über unsichere Netzwerke notwendig ist.

Ein klassisches Beispiel ist der Homeoffice- und Remote-Zugriff:

Mitarbeitende verbinden sich über ein VPN mit dem internen Firmennetzwerk und erhalten dadurch Zugriff auf Server, Dateien oder Anwendungen, als säßen sie direkt im Büro. Dadurch bleiben vertrauliche Informationen geschützt, selbst wenn von unterwegs oder über ein öffentliches WLAN gearbeitet wird.

Auch im Non-Profit- und NGO-Bereich sind VPNs ein unverzichtbares Werkzeug. Sozialarbeiter, Missionsteams oder

Hilfsorganisationen nutzen verschlüsselte Verbindungen, um auf geschützte Systeme und sensible Daten zuzugreifen, etwa auf medizinische Informationen, interne Fallakten oder Kommunikationsplattformen. So lassen sich Einsätze sicher koordinieren, selbst in Regionen mit unzuverlässiger oder überwachter Infrastruktur.

Besonders in der Seelsorge und Menschenrechtsarbeit spielen VPNs eine lebenswichtige Rolle. Christen, Aktivisten und Journalisten in sensiblen oder autoritären Regionen nutzen sie, um anonymisiert und geschützt kommunizieren zu können. Durch den verschlüsselten Tunnel werden ihre Aktivitäten verschleiert und die Identität der Beteiligten geschützt. Ein Akt der digitalen Nächstenliebe, bei dem Technik zu einem Werkzeug des Friedens und der Wahrung von Menschenwürde wird.

Ein weiteres wichtiges Einsatzfeld ist die Absicherung von IoT-Geräten. Netzwerkkameras, Sensoren oder kleine Server werden so konfiguriert, dass sie ausschließlich über ein VPN erreichbar sind. Dadurch bleiben sie vor direktem Zugriff aus dem Internet geschützt, und Administratoren können sicher auf sie zugreifen, ohne das Risiko offener Ports oder unsicherer Fernzugänge.

Ob privat, beruflich oder gemeinnützig VPNs schaffen Vertrauen in digitalen Räumen, wo es sonst leicht verloren gehen könnte.

VPN-Arten & Tunneltechniken

VPNs können in unterschiedlichen Szenarien eingesetzt werden, vom gesicherten Zugriff eines Heimarbeitsplatzes bis hin zur Umgehung von Zensur und zum Schutz in repressiven Regimen und zur verschlüsselten Kommunikation zwischen zwei Rechenzentren. Dieses Kapitel bietet eine strukturierte Übersicht über die verschiedenen Arten von VPN-Verbindungen, die zugrunde liegenden Tunneltechniken und deren technische Einordnung.

VPN-Tunnel
Der geschützte Datenkanal zwischen zwei Netzwerken oder einem Client und einem Server, über den Daten verschlüsselt und authentifiziert übertragen werden.

VPN-Arten im Überblick

Art	Beschreibung	Beispiel
Remote Access VPN	Verbindung einzelner Clients mit einem zentralen Netzwerk	Homeoffice-User verbindet sich mit der Firmenzentrale
Site-to-Site VPN	Verbindung ganzer Netzwerke miteinander	Zwei Filialen eines Unternehmens
Client-Based VPN	VPN über spezielle Software oder integrierte OS-Funktion (inkl. Funktionen wie Kill-Switch oder DNS-Leak-Schutz)	OpenVPN, WireGuard, Cisco AnyConnect
Cloud-/Mesh-VPN	Verbindung über cloudbasierte oder dezentrale Strukturen	Tailscale, ZeroTier

Tunnelprotokolle & Technologien

Protokoll	Layer	Verschlüsselung	Bemerkung
PPTP	Layer 2	schwach (MS-CHAPv2)	Veraltet, unsicher
L2TP/IPsec	Layer 2/3	stark (IPsec)	Doppelt gekapselt, mäßige Performance
OpenVPN	Layer 3 (TUN), Layer 2 (TAP)	stark (TLS)	Flexibel, etabliert
WireGuard	Layer 3	Modern (Curve25519, ChaCha20)	Schnell, kompakt, im Kernel; verwendet ausschließlich schlüsselbasierte Peer-Authentifizierung, keine integrierte Benutzerverwaltung oder dynamische IP-Zuweisung; zusätzliche Infrastruktur erforderlich.
IKEv2/IPsec	Layer 3	stark (IPsec)	Robust für Mobilgeräte

Layer-Zuordnung (TUN/TAP)

TUN/TAP

Virtuelle Netzwerkgeräte. TUN wird typischerweise für IP-Pakete auf Layer 3 (Routing) verwendet, während TAP bei der Übertragung von Ethernet-Frames auf Layer 2 (Bridging) zum Einsatz kommt.

- **TUN:** Layer 3 – überträgt IP-Pakete (Routing-basiert)
- **TAP:** Layer 2 – überträgt Ethernet-Frames (Bridge-basiert)

Vergleich

TUN → geringere MTU-Probleme, effizienter, jedoch ungeeignet für Anwendungen mit Layer-2-Anforderungen wie NetBIOS oder bestimmte Spiele

TAP → für komplexe Netze mit DHCP/Broadcast notwendig

Technologievergleich – Sicherheit & Leistung

Kriterium	OpenVPN	WireGuard	IKEv2/IPsec	L2TP/IPsec
Sicherheit	Hoch	Sehr hoch	Hoch	Hoch, *wenn korrekt konfiguriert* Mittel, wegen PSK, NAT-Traversal, Legacy-Clients
Geschwindigkeit	Mittel	Hoch	Hoch	Niedrig

Kriterium	OpenVPN	WireGuard	IKEv2/IPsec	L2TP/IPsec
Einrichtung	Flexibel	Sehr einfach	Komplex	Mittel
Mobilgeräte-Support	Gut	Sehr gut	Sehr gut	Schwach
Portabilität	Hoch	Hoch	Mittel	Mittel

Protokollwahl nach Einsatzzweck

Anwendungsfall	Empfohlenes Protokoll
Homeoffice mit Windows-Clients	OpenVPN oder IKEv2/IPsec
Mobile Nutzer (iOS, Android)	WireGuard oder IKEv2/IPsec
Ressourcenschonende Verbindungen	WireGuard
Legacy-Kompatibilität	L2TP/IPsec (nur wenn nötig)

Auch bei der Wahl der Technologie zeigt sich Verantwortung

Ein sicheres Protokoll schützt nicht nur Technik, sondern Menschen. In verfolgten Regionen kann ein sicheres VPN Lebensschutz bedeuten. Transparenz & Offenheit (Open Source) reflektieren Werte wie Integrität und Demut.

Die Wahl des richtigen VPN-Typs und Protokolls ist abhängig vom Einsatzzweck, der Umgebung und den ethischen Grundhaltungen. Wer Technologie bewusst wählt, schützt nicht

nur Daten, sondern schafft Vertrauen und ermöglicht sichere, freie Kommunikation über Grenzen hinweg.

Merksatz: *Ein Tunnel verbindet nicht nur Netze, sondern auch Menschen in Verantwortung.*

VPN-Grundlagen: Begriffe & Protokolle

Um VPNs effektiv einsetzen und sicher konfigurieren zu können, ist ein solides technisches Grundverständnis unverzichtbar. Dieses Kapitel klärt zentrale Begriffe, erläutert die wichtigsten Netzwerkprotokolle und zeigt, auf welcher Schicht des OSI-Modells VPN-Komponenten arbeiten. Ebenso wird auf Encapsulation, Portnutzung und die Unterscheidung zwischen Authentifizierung, Integrität und Verschlüsselung eingegangen.

Encapsulation
Umschließen eines Datenpakets mit einem zusätzlichen Header zur Weiterleitung z. B. ein IP-Paket wird in ein UDP-Paket eingekapselt, wie es bei OpenVPN der Fall ist, um es sicher über ein anderes Netzwerk transportieren zu können.

Wichtige Begriffe im VPN-Kontext

Begriff	Bedeutung
Client	Das Gerät oder die Software, die sich mit dem VPN verbindet
Server	Der Gegenpart, der die Verbindung annimmt und absichert
Tunnel	Die verschlüsselte Verbindung zwischen Client und Server
Peer	Kommunikationspartner in einer VPN-Architektur (z. B. WireGuard)
Gateway	Netzwerkpunkt, der Datenverkehr weiterleitet oder absichert
Encapsulation	Das Verpacken eines Protokolls in ein anderes

Begriff	Bedeutung
Split Tunneling	Nur bestimmter Traffic läuft über das VPN, der Rest direkt – z. B. Streaming-Dienste oder Druckeranfragen können direkt, ohne VPN, geleitet werden.

Wichtige Protokolle & Ports

Protokoll	Aufgabe	Standardport
UDP	Schnelles verbindungsloses Transportprotokoll	z. B. 1194 (OpenVPN), 51820 (WireGuard)
TCP	Verbindungsorientiert, fehlerkorrigierend	z. B. 443 für OpenVPN über TLS
IP	Unterschicht für Adressierung und Routing	–
TLS	Transport Layer Security für verschlüsselte Sessions – wird häufig als Transportabsicherung für OpenVPN eingesetzt	443
IPsec	Verschlüsselung auf IP-Ebene	Protokolle 50 (ESP) & 51 (AH) (AH wird in der Praxis selten genutzt)

OSI-Modell & VPN-Zuordnung

OSI-Schicht	VPN-Komponenten
Schicht 7 (Anwendung)	VPN-Software-Clients

OSI-Schicht	VPN-Komponenten
Schicht 4 (Transport)	TCP/UDP für OpenVPN, WireGuard etc.
Schicht 3 (Netzwerk)	IPsec, WireGuard (IP-basiert)
Schicht 2 (Sicherung)	L2TP, TAP-Interface

Hinweis *Die tatsächliche Schicht hängt vom verwendeten Tunnelprotokoll ab (z. B. OpenVPN kann sowohl Layer 2 als auch Layer 3 sein).*

VPN-in-VPN & Encapsulation

In besonders sensiblen Umgebungen kann es sinnvoll sein, mehrere VPN-Verbindungen ineinander zu verschachteln – ein Prinzip, das als VPN-in-VPN oder Encapsulation bezeichnet wird. Dabei wird ein VPN-Tunnel innerhalb eines anderen aufgebaut, zum Beispiel ein schneller WireGuard-Tunnel, der in einen TLS-gesicherten OpenVPN-Tunnel eingebettet ist. Auf diese Weise entstehen zusätzliche Sicherheitsschichten, die sowohl den Inhalt als auch die Art des verwendeten Protokolls verschleiern.

Der Vorteil einer solchen Kaskadierung liegt in der erhöhten Anonymität und Protokollverschleierung. Selbst wenn der äußere Tunnel erkannt oder analysiert wird, bleibt der innere Datenstrom verborgen. Das kann in Hochsicherheitsumgebungen, bei sensiblen politischen oder journalistischen Einsätzen oder in Ländern mit massiver Netzüberwachung sinnvoll sein. Gleichzeitig steigt jedoch der technische Aufwand: Jede zusätzliche Tunnelung erhöht den Overhead, verringert die Übertragungsgeschwindigkeit und erfordert sorgfältige Abstimmung der MTU-Werte (Maximum Transmission Unit), damit keine Paketfragmentierungen auftreten.

Auch die Firewallregeln müssen angepasst werden, damit beide VPN-Schichten reibungslos miteinander arbeiten. Werden Protokolle wie UDP und TCP gemischt oder in unterschiedlicher Reihenfolge verschachtelt, kann es sonst zu Verbindungsabbrüchen oder unerwarteten Latenzen kommen.

Richtig konfiguriert bietet ein VPN-in-VPN-Szenario eine außergewöhnlich hohe Sicherheit und Erkennungsresistenz – es sollte jedoch nur dort eingesetzt werden, wo die zusätzliche Komplexität tatsächlich einen Mehrwert bringt.

MTU-Diagnose (Kurzrezept):
Unter Linux kann man MTU/Fragmentierungsprobleme testen mit:
`ping -M do -s 1372 <vpn-gateway>`
Wenn Pakete fragmentiert werden, sukzessive -s verkleinern bis Antwort kommt. Daraus tun-mtu/mssfix ableiten (z. B. tun-mtu 1400, mssfix 1360) je nach VPN-Technologie.

Authentifizierung, Integrität, Verschlüsselung

Konzept	Zweck	Beispiel
Authentifizierung	Sicherstellen, dass Nutzer & Server echt sind	Zertifikat, Pre-Shared Key
Integrität	Schutz vor Manipulation	HMAC, Signaturen
Verschlüsselung	Schutz vor Mitlesen	AES, ChaCha20

Beispiel: Paketfluss bei OpenVPN (TUN)

1. Anwendung sendet IP-Paket an Zieladresse
2. OpenVPN verpackt IP-Paket in UDP/TCP mit TLS
3. Paket wird an Server gesendet → entschlüsselt → entpackt
4. Server leitet das Originalpaket ins Zielnetz weiter

VPN-Konfiguration Ports Dateien & Netzwerkschnittstellen

Dieses Kapitel gibt einen praxisnahen Einblick in die typischen Konfigurationsdateien, Netzwerkparameter und Ports, die beim Aufbau und Betrieb eines VPNs zum Einsatz kommen. Es werden exemplarische Konfigurationsdateien für OpenVPN und WireGuard vorgestellt, Begriffe wie `tun0`, `eth0`, `dev` und `proto` erklärt und gezeigt, wie sich Netzwerkschnittstellen, Routing und Firewall in das Gesamtbild einfügen.

Wichtige Konfigurationsparameter

Parameter	Bedeutung	Beispiel
`dev`	Virtuelle Netzwerkschnittstelle (Layer 3: tun, Layer 2: tap)	`tun0`, `tap0`
`proto`	Transportprotokoll – UDP ist schneller, TCP robuster hinter Firewalls	`udp`, `tcp`
`port`	Port, auf dem der VPN-Dienst lauscht	`1194` (OpenVPN), `51820` (WireGuard)
`remote`	Zieladresse des Servers	`vpn.example.com`
`ifconfig` / `ip`	IP-Konfiguration (Legacy/Modern) – Hinweis: `ifconfig` gilt als veraltet, `ip` wird empfohlen	`10.8.0.1 10.8.0.2`
`AllowedIPs`	Erlaubte IP-Ziele bei WireGuard	`0.0.0.0/0`, `::/0`

OpenVPN: Beispielkonfiguration (Server) [1]

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
verb 3
```

OpenVPN: Beispielkonfiguration (Client)

```
dev tun
proto udp
remote vpn.example.com 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
remote-cert-tls server
verb 3
```

[1] dh dh.pem ist korrekt, aber in neueren OpenVPN-Setups oft durch ECDH ersetzt.

WireGuard: Beispielkonfiguration (Server) [2]

```
[Interface]
PrivateKey = AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Address = 10.0.0.1/24
ListenPort = 51820

[Peer]
PublicKey = BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
AllowedIPs = 10.0.0.2/32
```

WireGuard: Beispielkonfiguration (Client)

```
[Interface]
PrivateKey = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Address = 10.0.0.2/32
DNS = 10.0.0.1

[Peer]
PublicKey = YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Endpoint = vpn.example.com:51820
AllowedIPs = 0.0.0.0/0, ::/0
PersistentKeepalive = 25
```

Netzwerkschnittstellen im VPN-Kontext

Schnittstelle	Zweck
`tun0`	Layer-3 Tunnel (IP) – Standard bei OpenVPN, WireGuard
`tap0`	Layer-2 Tunnel (Ethernet) – für Broadcast, DHCP
`eth0`	Physische Netzwerkkarte des Hosts
`wg0`	WireGuard Interface (virtuell); WireGuard nutzt ebenfalls Layer 3 (IP)

[2] Achtung: Dieser Port muss in der Firewall explizit freigegeben werden

Zusammenhang: Interface ↔ Routing ↔ Port

Damit ein VPN stabil und sicher funktioniert, müssen die drei zentralen Komponenten Interface, Routing und Port korrekt aufeinander abgestimmt sein. Der Parameter dev tun0 erzeugt dabei die virtuelle Netzwerkschnittstelle, über die der gesamte VPN-Datenverkehr läuft. Diese Schnittstelle fungiert als logischer Tunnel, über den alle verschlüsselten Pakete transportiert werden.

Der Port legt fest, über welchen Eingangspunkt der Server Verbindungen entgegennimmt. In den meisten Fällen wird hierfür Port 1194 genutzt, der in der Konfiguration mit proto udp definiert ist. Dieser Eintrag bestimmt zugleich das Transportprotokoll, UDP in diesem Fall, und sorgt dafür, dass der Server eingehende Anfragen auf genau diesem Port überwacht und verarbeitet.

Über passende Routingregeln wird anschließend gesteuert, welche IP-Pakete durch das VPN-Interface geleitet werden. Das System legt also fest, ob nur bestimmte Subnetze (beim Split-Tunneling) oder der gesamte Verkehr (beim Full-Tunneling) über tun0 laufen.

Eine entscheidende Rolle spielt dabei die Firewall, sei es durch iptables unter Linux oder durch grafische Verwaltungsoberflächen wie OPNsense. Sie kontrolliert, ob Verbindungen auf den definierten Port erlaubt oder blockiert werden und schützt so den Tunnel vor ungewollten Zugriffen. Erst wenn diese vier Elemente, Interface, Port, Routing und Firewall, sauber ineinandergreifen, entsteht ein sicherer und funktionaler VPN-Kanal.

Platzhalter & Sonderzeichen in Konfigs

Zeichen	Bedeutung
`0.0.0.0/0`	Alle IPv4-Ziele
`::/0`	Alle IPv6-Ziele
`%i`	Interface-abhängige Platzhalter (SystemD)
`#`	Kommentar (OpenVPN)
`;`	Alternativer Kommentar (WireGuard, ini-Stil)

Was passiert beim Start eines VPN-Dienstes?

1. Das System lädt die Konfigurationsdatei
2. Es wird geprüft, ob Schlüssel/Zertifikate vorhanden sind
3. Die virtuelle Schnittstelle (`tun0`, `wg0`) wird erstellt
4. Firewall, NAT und Routingregeln werden gesetzt
5. Der Dienst bindet an den definierten Port und lauscht
6. Bei eingehender Verbindung startet die Aushandlung (Handshake)

Die VPN-Konfiguration ist das Herzstück eines sicheren Tunnels. Nur wer die Parameter versteht, kann Fehlkonfigurationen vermeiden und Sicherheit sowie Stabilität gewährleisten. Dieses Kapitel liefert die Grundlagen für genau diese Kompetenz – mit Praxisbezug, Tiefe und ethischem Blick.

Merksatz: *Ein sicherer Tunnel beginnt mit einer klaren Konfiguration*.

VPN-Nutzung: Windows Linux macOS Mobilgeräte

Dieses Kapitel beleuchtet die praktische Nutzung von VPN-Verbindungen auf verschiedenen Betriebssystemen. Je nach Plattform unterscheidet sich die Installation, Einrichtung und Verwaltung von VPNs erheblich. Sowohl grafische Benutzeroberflächen als auch Kommandozeilen-Tools kommen zum Einsatz. Ergänzt wird dies durch Hinweise zu Fehlerbehebung, Sicherheit und einem christlich-ethischen Blick auf verantwortungsvollen Umgang mit Technik.

Windows

Nutzung und Einrichtung

Unter Windows stehen mehrere Wege zur Verfügung, um eine VPN-Verbindung einzurichten und zu betreiben. Am geläufigsten ist die Konfiguration über das „Netzwerk- und Freigabecenter“, wo eine neue VPN-Verbindung erstellt und anschließend über die Windows-GUI verwaltet wird. Alternativ lässt sich der Verbindungsaufbau auch über den älteren Einwahlmechanismus mit `.pbk`-Dateien und dem Verbindungsmanager (`RASDIAL`) steuern, was sich für Skripte oder automatisierte Abläufe eignet. In vielen Szenarien kommen zudem Drittanbieter-Clients zum Einsatz, insbesondere die OpenVPN GUI oder der WireGuard-Client, die jeweils ihre eigenen Profile und Optionen mitbringen.

Beispiel OpenVPN GUI

Für den Betrieb der OpenVPN GUI müssen die Konfigurationsdateien korrekt abgelegt und benannt sein. Üblich ist der Pfad `%ProgramFiles%\OpenVPN\config`; dort liegende .ovpn-Dateien werden von der GUI erkannt und zur Auswahl

angeboten. Damit Routen gesetzt, Schnittstellen konfiguriert und Firewall-Ausnahmen angewendet werden können, sollte die OpenVPN GUI mit Administratorrechten gestartet werden. Außerdem ist ein passender virtueller Netzwerkadapter erforderlich: Der TAP-Windows-Adapter muss installiert und funktionsfähig sein, damit der Tunnel als Schnittstelle bereitgestellt werden kann.

Häufige Fehlerquellen

Probleme entstehen unter Windows häufig durch Treiberkonflikte rund um TAP/TUN-Adapter, insbesondere nach größeren Updates oder paralleler Installation mehrerer VPN-Lösungen. Ebenso blockiert die Windows-Firewall nicht selten die für den Dienst benötigten UDP- oder TCP-Ports, was den Handshake verhindert oder die Verbindung instabil macht. Ein dritter typischer Stolperstein sind fehlende Administratorrechte: Ohne erhöhte Rechte können Routen nicht gesetzt und Schnittstellen nicht korrekt konfiguriert werden, sodass zwar ein scheinbarer Verbindungsaufbau stattfindet, der Datenverkehr jedoch nicht über den Tunnel läuft.

Linux

Nutzung und Einrichtung

Unter Linux gibt es drei gängige Wege zum VPN: über den NetworkManager (grafisch oder per `nmcli`), direkt über die CLI mit OpenVPN (z. B. `sudo openvpn client.conf`) und mit WireGuard via `sudo wg-quick up wg0`. Der NetworkManager eignet sich für dauerhafte Profile und Policy-basierte Einstellungen; die CLI ist ideal für Tests, Skripte und minimale Systeme. Für WireGuard übernimmt `wg-quick` das Laden der Schlüssel, Routen und (optional) DNS-Einträge in einem Schritt.

Dateipfade

OpenVPN-Clientdateien liegen systemweit typischerweise unter `/etc/openvpn/client/`, benutzerspezifische Profile unter `~/.config/openvpn/`. WireGuard-Konfigurationen befinden sich in der Regel in /etc/wireguard/wg0.conf.

Dienstintegration & DNS

Damit Verbindungen automatisch starten, kannst du systemd aktivieren, z. B. mit `systemctl enable openvpn-client@client`. Bei WireGuard erledigt `wg-quick@wg0.service` das gleiche. DNS erfordert oft eine explizite Vorgabe: Entweder per resolvconf/openresolv, über den NetworkManager (der `systemd-resolved` korrekt füttert), oder direkt in `wg0.conf` via `DNS=`-Eintrag. Wichtig ist, dass die Namensauflösung über den Tunnel geht, sonst drohen DNS-Leaks.

Typische Fehlerquellen

Häufig sind MTU-Probleme: Der Tunnel steht, aber es fließt kein Traffic oder nur manche Seiten laden. Abhilfe: MTU gezielt setzen (z. B. WireGuard `MTU=1420` in `wg0.conf`; OpenVPN ggf. `mssfix` passend wählen) und neu testen. Ebenfalls verbreitet: DNS-Leaks durch falsch angebundenes `systemd-resolved`. Prüfe, ob die VPN-Schnittstelle als *Link-specific DNS* hinterlegt ist und externe Resolver außerhalb des Tunnels blockiert werden.

macOS

Nutzung und Einrichtung

macOS bietet bereits systemseitig eine native Unterstützung für VPN-Verbindungen in den Systemeinstellungen, wobei die Protokolle L2TP über IPsec und IKEv2 direkt integriert sind. Diese

Variante eignet sich besonders für einfache Unternehmens- oder Bildungsnetzwerke, bei denen der Zugriff zentral bereitgestellt wird. Für flexiblere und modernere VPN-Lösungen wie OpenVPN oder WireGuard stehen Drittanbieter-Tools wie Tunnelblick oder die offizielle WireGuard-App für macOS zur Verfügung.

Beispiel: Tunnelblick (OpenVPN)

Das Programm Tunnelblick integriert OpenVPN nahtlos in macOS und bringt eine benutzerfreundliche Oberfläche für Verbindungsaufbau und -verwaltung mit. Die Konfigurationsdateien liegen standardmäßig im Verzeichnis `/Library/Application Support/Tunnelblick/Configurations`, wo sie nach dem Start automatisch erkannt werden. Beim Aufbau der Verbindung ist eine Administratorbestätigung erforderlich, da Tunnelblick Systemkomponenten verändert (Netzwerkschnittstellen und Routen).

Hinweise & Fehlerquellen

In aktuellen macOS-Versionen blockiert Apple teilweise unsignierte TUN/TAP-Treiber, wodurch ältere VPN-Clients nicht mehr funktionieren. Tunnelblick umgeht dieses Problem, indem es eigene signierte Treiber mitliefert. Alternativ können manuell installierte, signierte Drittanbietertreiber helfen.

Ebenfalls wichtig ist die Kontrolle der Firewall (pf). Sie kann eingehende oder ausgehende VPN-Verbindungen blockieren, wenn keine passenden Regeln gesetzt sind. In der Praxis sollte geprüft werden, ob die VPN-Ports (z. B. 1194/UDP für OpenVPN) freigegeben sind.

Sleep- und Wake-Zustände

Wenn der Mac in den Ruhezustand versetzt wird, unterbricht das System automatisch die Netzwerkverbindungen, der VPN-Tunnel wird dabei oft getrennt und muss nach dem Aufwachen manuell oder per Skript neu aufgebaut werden. Wer auf stabile Dauerverbindungen angewiesen ist, sollte diese Eigenheit bei der Planung berücksichtigen und ggf. Automatisierungen für den Wiederaufbau integrieren.

Mobilgeräte (Android & iOS)

Nutzung und Besonderheiten

VPNs lassen sich auch auf mobilen Geräten komfortabel und sicher nutzen. Sowohl Android als auch iOS bieten native Unterstützung sowie eine breite Auswahl an offiziellen Clients, etwa OpenVPN Connect oder die WireGuard-App. Die Einrichtung ist in der Regel unkompliziert: Eine bereitgestellte `.ovpn`- oder `.conf`-Datei kann per QR-Code, Dateifreigabe oder direkt über die App importiert werden. Danach genügt ein Fingertipp, um die Verbindung herzustellen.

Android

Auf Android-Geräten hängt die Stabilität einer VPN-Verbindung stark vom Hersteller und der jeweiligen Systemversion ab. Einige Geräte verfügen über aggressive Stromsparmodi, die Hintergrundverbindungen nach kurzer Inaktivität trennen, darunter auch VPN-Tunnel. Um eine dauerhafte Verbindung zu gewährleisten, sollte der VPN-Client von Energiesparmaßnahmen ausgenommen und mit den nötigen Berechtigungen für Hintergrundaktivität ausgestattet werden. Nur Apps mit dauerhaftem Hintergrundzugriff können eine stabile, kontinuierliche VPN-Verbindung aufrechterhalten.

iOS

Unter iOS ist die VPN-Integration tief ins System eingebettet, was die Nutzung besonders einfach, aber auch stärker reglementiert macht. Manche Mobilfunkanbieter blockieren allerdings UDP-Traffic, wodurch VPN-Protokolle wie OpenVPN beeinträchtigt werden können. In solchen Fällen bietet sich der Wechsel auf TCP-basierte Verbindungen oder auf Protokolle wie IKEv2 an, die von iOS nativ unterstützt werden.

Für Unternehmen oder Organisationen bietet Apple mit dem Apple Configurator und MDM-Systemen (Mobile Device Management) die Möglichkeit, VPN-Profile zentral zu verteilen und abzusichern. So lassen sich gerätespezifische Richtlinien, Zertifikate und Verbindungsparameter automatisiert ausrollen, ein wichtiger Schritt, um Sicherheit und Konsistenz in größeren Umgebungen zu gewährleisten.

Sicherheit & Besonderheiten je OS

Betriebssystem	Besonderheiten / Hinweise
Windows	Adminrechte nötig, Routen oft fehleranfällig
Linux	Sehr flexibel, aber CLI-lastig, `systemd` beachten
macOS	Treibersupport schwächer, GUI empfohlen
Android/iOS	App-Stabilität hängt stark von Systemrichtlinien ab

VPNs lassen sich auf allen modernen Plattformen sicher betreiben, doch jedes Betriebssystem bringt seine Eigenheiten mit. Wer diese kennt, kann Verbindungen stabil und vertrauenswürdig konfigurieren.
Merksatz: *Ein Tunnel schützt nur dann, wenn beide Enden verstehen, was sie tun*

Plattform- & Clientkonfiguration

Plattformübergreifende VPN-Nutzung ist essenziell in modernen IT-Umgebungen. Die Konfiguration unterscheidet sich je nach Betriebssystem, doch gewisse Prinzipien wie Zertifikate, Routen oder Autostartmechanismen gelten überall. Dieses Kapitel erläutert Konfigurationen für Windows, Linux, macOS und mobile Systeme, inklusive praktischer `.ovpn`-Beispiele.

Windows-Client (GUI & CLI)

GUI (OpenVPN GUI)

Installation: openvpn.net/download-open-vpn/
`.ovpn`-Datei in `C:\Program Files\OpenVPN\config` speichern
Rechtsklick → „Als Administrator ausführen“
Zertifikatsabfragen bestätigen

CLI (Windows Terminal oder cmd)

`openvpn --config client.ovpn`
(erfordert Administratorrechte unter Windows; der Befehl muss i m Konfigurationsverzeichnis der .ovpn-Datei ausgeführt werden)

Linux-Client (CLI, Installation & Autostart)

Installation (Debian/Ubuntu)

```
sudo apt update && sudo apt install openvpn -y
```

CLI-Nutzung[3]

[3] Hinweis: Der Befehl muss im Verzeichnis mit der .ovpn-Datei ausgeführt werden oder es muss ein absoluter Pfad verwendet werden

```
sudo openvpn --config /etc/openvpn/client/client.ovpn
```

Autostart via `systemd` (z. B. Debian/Ubuntu)

```
sudo cp client.ovpn /etc/openvpn/client/client.conf
sudo systemctl enable openvpn-client@client
sudo systemctl start openvpn-client@client
```

macOS-Client (Tunnelblick & Alternativen)

Tunnelblick [4]

- Download: tunnelblick.net
- `.ovpn`-Datei doppelklicken → Import
- Verbindung starten aus Menüleiste

Mobile Clients (Android & iOS)

Android (z. B. OpenVPN for Android) [5]

- `.ovpn`-Datei via Dateimanager importieren
- Zertifikate ggf. getrennt angeben
- Energiesparmodus deaktivieren oder App whitelisten

iOS (OpenVPN Connect)

- `.ovpn` via E-Mail oder iCloud importieren
- iOS-Profil ggf. anpassen
- Hintergrundverbindung aktivieren

[4] **Alternative**: Viscosity (kostenpflichtig)

[5] **Achtung:** Bestimmte Android-Versionen nutzen aggressive Doze-Modi, die VPN-Verbindungen automatisch trennen können.

Beispiel Client.ovpn (universell)[6] [7]

```
client
dev tun
proto udp
remote vpn.example.org 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
tls-auth ta.key 1
cipher AES-256-GCM
auth SHA256
remote-cert-tls server
verb 3
```

Besonderheiten bei Autostart & Energiemanagement

- **Windows:** „Taskplaner“ oder Autostart-Ordner nutzen
- **Linux:** systemd bevorzugen, `rc.local` als Notfalllösung
- **macOS:** Tunnelblick erlaubt automatisches Starten bei Login
- **Android/iOS:** Stromsparoptionen verhindern oft stabile VPN-Verbindungen

Beispiel server.conf (Auszug für OpenVPN-Server)

```
dev tun
proto udp
port 1194
```

[6] Diese Option sorgt dafür, dass beim Reconnect vorhandene Schlüsseldateien nicht neu geladen werden müssen

[7] Hält das virtuelle Netzwerkinterface aktiv, wichtig für stabile Wiederverbindungen

```
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
dh /etc/openvpn/dh.pem
tls-auth /etc/openvpn/ta.key 0
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 1.1.1.1"
keepalive 10 120
cipher AES-256-GCM
auth SHA256
persist-key
persist-tun
status /var/log/openvpn-status.log
verb 3
```

Erklärung der .ovpn-Parameter

OpenVPN-Konfigurationsdateien steuern, wie sich ein Client mit dem VPN-Server verbindet. Sie enthalten Parameter zur Authentifizierung, Verschlüsselung, Netzwerkrouting und Logging. Eine sinnvolle Konfiguration verbessert Stabilität und Sicherheit erheblich.

Parameter und ihre Bedeutung

Parameter	Bedeutung
`client`	Aktiviert Client-Modus
`server`	Aktiviert Server-Modus mit Standardparametern
`dev tun` / `dev tap`	Wählt Netzwerkgerät: TUN (IP) oder TAP (Ethernet)
`proto udp` / `proto tcp`	UDP oder TCP als Transportprotokoll

Parameter	**Bedeutung**
`remote`	Zieladresse & Port des Servers
`port`	Port, auf dem der Server lauscht (Server-seitig)
`resolv-retry infinite`	Endloser Verbindungsversuch bei Fehlern
`nobind`	Verwendet keinen festen lokalen Port (Client-seitig)
`persist-key`	Behalte Schlüsseldateien beim Reconnect
`persist-tun`	TUN-Gerät bleibt erhalten beim Reconnect
`ca, cert, key`	Pfade zu Zertifikatsdateien
`dh`	Pfad zur Diffie-Hellman-Datei (Server-seitig)
`tls-auth`	HMAC-Schutz vor Port-Scanning
`tls-crypt`	Verschlüsselt TLS-Handshake (ersetzt tls-auth)
`remote-cert-tls server`	Client prüft, ob Gegenstelle ein gültiger Server ist
`verify-client-cert`	Server prüft Client-Zertifikat (optional oder erforderlich)
`cipher`	Symmetrisches Verschlüsselungsverfahren (z. B. AES-256-GCM)
`auth`	Authentifizierungs-Hash (z. B. SHA256)
`compress / comp-lzo`	*Aktiviert Datenkompression (veraltet)*

Parameter	Bedeutung
`topology subnet`	Empfohlene Topologie für IPv4 Routing
`push`	Server schickt Client bestimmte Optionen (z. B. DNS, Routen)
`route`	Fügt spezifische Routen hinzu
`route-nopull`	Client ignoriert automatisch gepushte Routen
`explicit-exit-notify`	Benachrichtigt Server bei Verbindungsende (UDP)
`keepalive`	Sende regelmäßig Ping/Pong zur Verbindungsüberwachung
`duplicate-cn`	Erlaubt mehrere Clients mit gleichem Zertifikat (nicht empfohlen)
`user / group`	Serverprozess läuft unter anderem Benutzer/Konto
`log / log-append`	Logfile schreiben oder anhängen
`status`	Schreibt Statusdatei (für Überwachung/Skripting)
`verb`	Legt Log-Detailgrad fest (0 = still, 3 = normal, 9 = Debug)
`mute`	Unterdrückt sich wiederholende Logzeilen
`crl-verify`	Verwendet eine Sperrliste (Certificate Revocation List)
`client-config-dir`	Individuelle Konfigurationen pro Client (Server-seitig), z. B. unterschiedliche Routen je Client durch separate

Parameter	Bedeutung
	Konfigurationsdateien im angegebenen Verzeichnis Konfigurationen pro Client (Server-seitig)
`ifconfig-pool-persist`	Speichert zugewiesene IPs für Wiederverbindungen
`management`	Startet Verwaltungs-Interface (z. B. für GUI)

Best Practices & Ergänzende Optionen

```
log /var/log/openvpn.log
log-append /var/log/openvpn.log
explicit-exit-notify 1
pull-filter ignore "route-ipv6"
block-outside-dns
```

Option	Erklärung
`log /var/log/openvpn.log`	Protokolliert Ausgaben in Logdatei (wichtig für Fehlersuche)
`log-append /var/log/openvpn.log`	Fügt neue Logs am Ende der Datei an
`explicit-exit-notify 1`	Signalisiert Session-Ende bei UDP-Verbindungen (für sauberes Trennen)
`pull-filter ignore "route-ipv6"`	Ignoriert IPv6-Routen vom Server (Vermeidung fehlerhafter IPv6-Routen)
`block-outside-dns`	Windows: Verhindert DNS-Leaks außerhalb des VPN-Tunnels [8]

Wenn du möchtest, ergänze das Ganze später mit deiner eigenen Konfiguration oder einem individuellen

[8] unter macOS und Linux ohne Wirkung

Kommentarbereich. Es lohnt sich, seine `.ovpn`-Dateien gut zu dokumentieren – gerade bei Fehlersuche oder Support.

Alternative: Login via `auth-user-pass` – besonders verbreitet in Unternehmensnetzwerken mit zentraler Authentifizierung wie RADIUS oder LDAP

```
client
dev tun
proto udp
remote vpn.example.org 1194
auth-user-pass
tls-auth ta.key 1
ca ca.crt
cipher AES-256-GCM
auth SHA256
verb 3
```

Wird häufig in Kombination mit Radius- oder PAM-Backends verwendet.

tls-auth vs. tls-crypt

Feature	**`tls-auth`**	**`tls-crypt`**
Funktion	HMAC-Schutzkanal	Verschlüsselt gesamtes TLS-Hello
Vorteil	Anti-Scan & Manipulationsschutz	Tarnung des VPN-Servers
Empfehlung	Gut	**Bevorzugt** (besserer Schutz)

Best Practices: Sichere .ovpn-Vorlage (Client)

```
client
dev tun
proto udp
remote vpn.example.com 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
tls-auth ta.key 1
remote-cert-tls server
cipher AES-256-GCM
auth SHA256
verb 3
explicit-exit-notify 1
block-outside-dns
pull-filter ignore "route-ipv6"
log /var/log/openvpn.log
```

Linux Desktop GUI – NetworkManager

```
nmcli connection import type openvpn file client.ovpn
nmcli connection up <Name>
```

Oder über: „Netzwerk → VPN → Importieren“ (Gnome/KDE)

Importmethoden iOS & Android (Details)

- **iOS:** via E-Mail, AirDrop, Dateien-App, iCloud
- **Android:** via Download-Ordner, Dateibrowser, „Teilen“ mit OpenVPN App
- **MDM:** Enterprise-Profile mit `.ovpn` Push

Die Plattformwahl beeinflusst die Clientkonfiguration, doch mit den richtigen `.ovpn`-Beispielen, Zertifikaten und systemgerechten Startmechanismen lässt sich jeder VPN-Client sicher und zuverlässig betreiben. Einheitliche Standards wie TLS und Zertifikatsauthentifizierung erleichtern dabei die Pflege.

Split-Tunneling & Dual Stack

Split-Tunneling erlaubt es, bestimmte Datenverbindungen durch das VPN zu schicken, während andere direkt ins Internet gehen. Das spart Bandbreite, erhöht aber auch das Risiko von Leaks.

Praxisbeispiel: Split-Tunneling unter Windows (WireGuard)

WireGuard selbst bietet kein eingebautes Split-Tunneling-Menü, aber du kannst es manuell konfigurieren.

Beispielkonfiguration [9]

```
[Interface]
PrivateKey = (dein privater Key)
Address = 10.0.0.2/32
DNS = 1.1.1.1

[Peer]
PublicKey = (Key vom Server)
Endpoint = vpn.example.com:51820
AllowedIPs = 10.0.0.0/24, 192.168.1.0/24
PersistentKeepalive = 25
```

Trick: Trage bei `AllowedIPs` *nur* die Subnetze ein, die durchs VPN gehen sollen (z. B. 10.0.0.0/24) – alles andere geht direkt über dein lokales Netz.
OpenVPN: Nutze `route-nopull` und `route`-Befehle in der `.ovpn`: [10]

```
route-nopull
route 10.0.0.0 255.255.255.0
route 192.168.10.0 255.255.255.0
```

[9] Auch IPv6-Netze möglich, z. B. `fd00::/8` für Dual-Stack-Setups
[10] Für IPv6 müssen zusätzlich `route-ipv6`-Einträge gesetzt werden.

Tabelle: Vorteile und Nachteile von Split-Tunneling

Vorteil	Nachteil
Reduziert Bandbreitenverbrauch	Risiko von DNS-Leaks
Lokale Geräte bleiben erreichbar	Split-Tunnel muss gut konfiguriert werden
Schnelleres Streaming/Surfen lokal	Sicherheitsüberwachung kann umgangen werden
Nur gezielter Traffic durch VPN	Nicht für Zero-Trust-Umgebungen geeignet

DNS-Leak-Warnung

Wenn nur der Datenverkehr durchs VPN geht, aber DNS-Anfragen lokal bleiben (z. B. an den Heimrouter), **können DNS-Leaks auftreten.**

Schutz:

- Nutze DNS in WireGuard (z. B. 1.1.1.1)
- Nutze block-outside-dns in OpenVPN (nur Windows; unter macOS und Linux ohne Wirkung) [11]
- Teste mit: https://www.dnsleaktest.com/

Häufige Fehler beim VPN-Tunnelaufbau – und wie du sie vermeidest

❌ **Dateien nicht gefunden – „No such file or directory“**

Fehlermeldung:
`Cannot open ta.key: No such file or directory`
Ursache:
Du befindest dich im falschen Ordner. OpenVPN findet die

[11] Das Verhalten hängt vom Client und Betriebssystem ab.