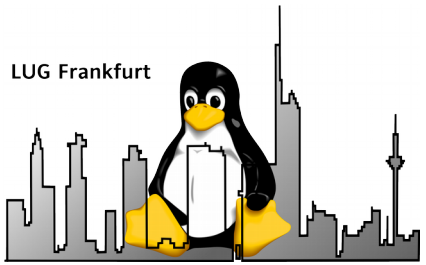
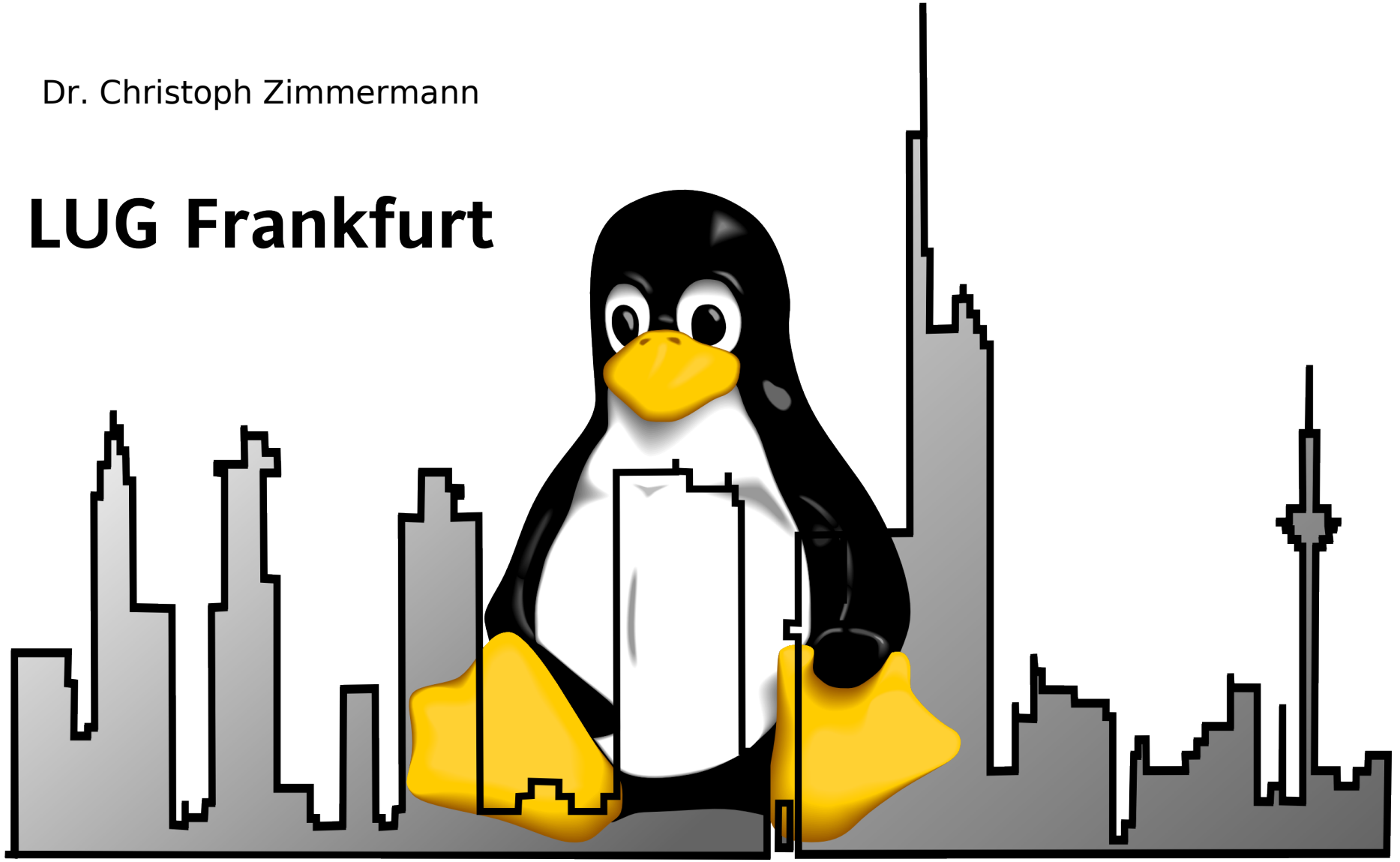


Anonymous & Freunde

Dr. Christoph Zimmermann

LUG Frankfurt



1. Warum?
2. Grundlagen
3. OpenVPN
4. Tor
5. Fortgeschrittenes
6. Diskussion / Fragen

Warum überhaupt Anonymisierung?

- Nicht nur Waffen, Drogen und Menschenhandel ☺
- Informationelle Selbstbestimmung
- Meinungsbildung und -äußerung (insbes. in politisch restriktiven Umgebungen)
- Recht auf Privatsphäre

Grundlagen

- Layer 3:
  - IPSec:
    - Kernel-Teil
    - Userland
- Layer 4:
  - OpenVPN
  - SSL / TLS
  - tcpcrypt

OpenVPN

- Freemium Software VPN Implementierung:
  - Community: wie wir's kennen
  - Access Server: inkl. propr. Erweiterungen: LDAP Integration, SMB Server, etc.
- Basierend auf OpenSSL + HMAC
- Authentifizierung:
  1. PSK
  2. Zertifikate
  3. Credentials
  4. Plugins: PAM, LDAP, etc.
- Tunneling via Routing-Tabellen-Modifikation

OpenVPN (ff.)

```
#
-------------------------------------
---------------------
# Air VPN | https://airvpn.org
# OpenVPN Client Configuration.
# AirVPN_United-Kingdom_TCP-80
#
-------------------------------------
---------------------

client
dev tun
proto tcp
remote gb.vpn.airdns.org 80
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
cipher AES-256-CBC
```

```
comp-lzo no
route-delay 5
verb 3
route-nopull
route www.rockantenne.de
<ca>
CA Zertifikat
</ca>
<cert>
Eigentliches Zertifikat zum
Verschlüsseln
</cert>
<key>
Privater Schlüssel für Zertifikat
</key>
```

VPN Tips

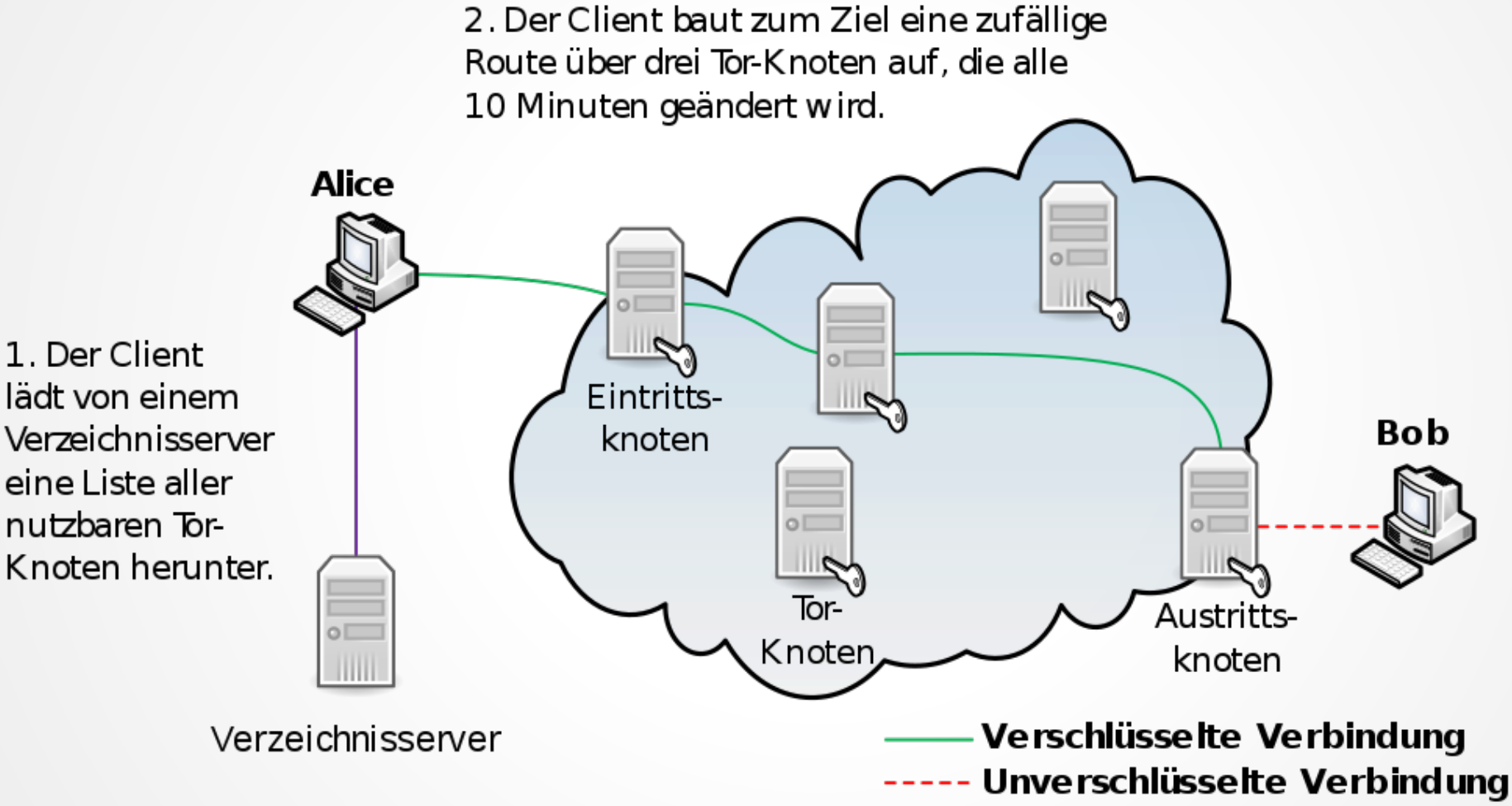
- Log-Policy
- Exit-Knoten
- Open Source Software oder binärer Klient?
- Port-Wahl: ISP-Throtteling
- Implementierung im Router: Abdeckung des gesamten Netzwerks (inkl. redundanter Anbieter)

Tor

- The Onion Router - ursprünglich US-Regierungsprojekt
- Virtuelle Verbindung mit verschachtelter Verschlüsselung der Inhalte inkl. Sender- und Empfänger IP-Adresse
- Implementierung durch spezielle Software oder auf SOCKS-Ebene:
  - Tor Browser Bundle (TBB): Firefox ESR mit eingebautem Tor-Client (Windows, OSX, Linux, BSD)

Tor (ff.)

- Architektur



Tor (ff.)

- Knoten:
  - Eintritt („entry guard“)
  - Relay
  - Exit
  - Brücke:
    - Relay, das nicht in Tor-Verzeichnissen gelistet ist
    - Muß evtl. manuell in Software konfiguriert werden (z. B. TBB)
    - Beispiel für einen „Pluggable Transport“ (nicht identifizierbarer Tor-Traffic zwischen Klient und ISP)

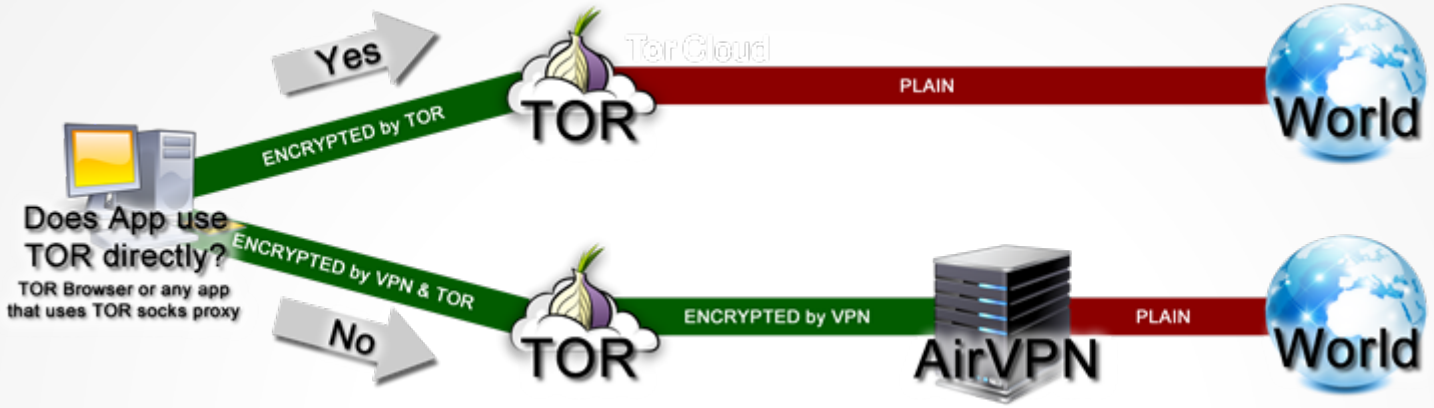
Tor-Tools

- TorBirdy: Thunderbird + Tor
- Orbot:
  - Tor-Klient für Android (lokaler HTTP-Proxy)
- Onion Browser:
  - Tor-Klient für iOS-Geräte
- Anonymizing Relay Monitor (ARM)
  - Python + Curses

.onion Sites

- Hidden Services; nur via Tor über „.onion“ erreichbar
- Addresse: 16 Zeichen alphanumerischer Hash (basierend auf öffentl. Schlüssel des Hidden Services)
- Konfiguration: in /etc/tor/torrc:
  - `HiddenServiceDir:` Speicherort für Schlüssel und Hash (wird vom Tor-Daemon angelegt)
  - `HiddenServicePort:` Umleitung des Dienstes auf Zielports

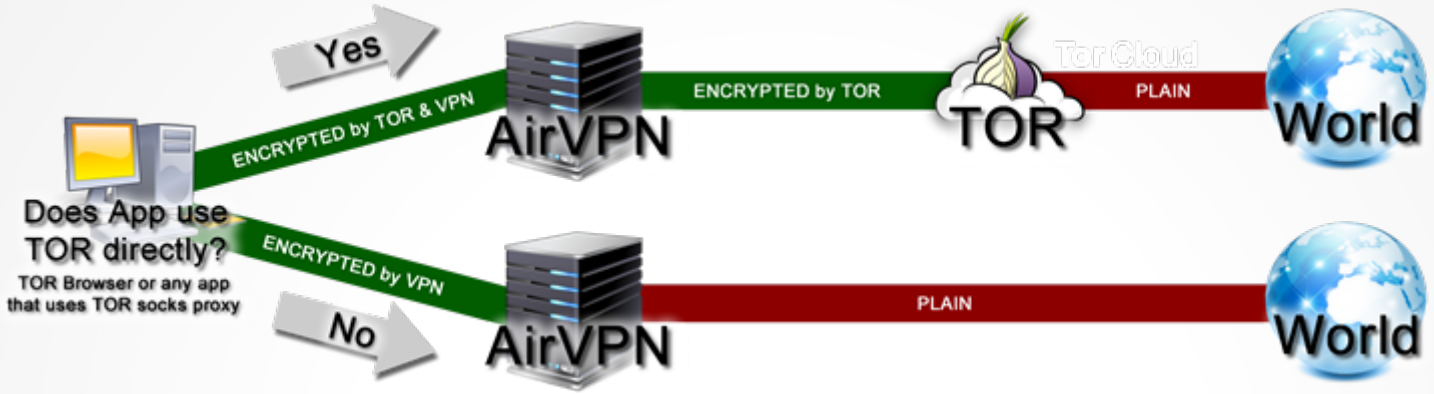
Fortgeschrittene Konfiguration



Tor vor VPN Knoten

VPN-Klient -> Tor SOCKS Klient	Geringfügige Leistungseinbußen durch doppelte Verschlüsselung (Software)
VPN sieht nur Exit-Knoten	
Traffic durch Exit-Knoten immer noch verschlüsselt	

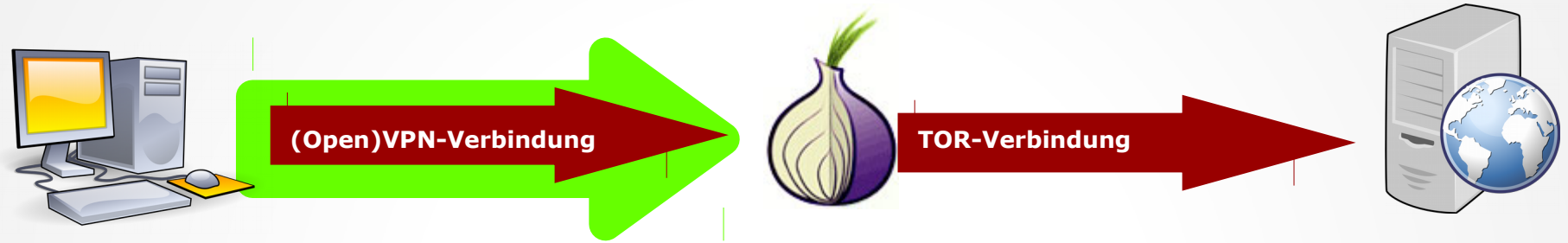
Fortgeschrittene Konfiguration (ff.)



VPN über TOR

Tor Klient -> VPN Klient	Kein Schutz vor kompromittierten Tor EXit Knoten
VPN sieht keine Zieladresse	VPN (Anbieter) muß Tor unterstützen
Tor-Entry Knoten sieht nur VPN-Adresse	

Super Stealth Mode :-)



.onion Knoten

VPN via .onion Knoten

Zutaten: (virtueller) Root-Server oder Container mit entsprechender Software

Zusätzlicher Schutz durch Kontrolle des .onion-Knotens

Privilegeskalation auf dem .onion-Knoten benötigt für VPN-Quelladresse (da log-Zugriff notwendig)

Tor: einige Angriffsvektoren

- Traffic-Analyse hinter Exit-Knoten (Gegenmaßnahme: Ende-zu-Ende TLS-Verschlüsselung)
- Übernahme der Kontrolle von 1/3 der Relays (theoretisch):
  - Akquisition der Schlüssel und Algorithmen-Basen (Entschlüsselung von 2 Schichten)
  - Rest: Statistischer Angriff auf verbleibende Schicht
- Verbindungs-Fingerabdruck (primär für .onion Sites):
  - Basierend auf Unterschied zwischen Traffic zu Onion- und normalen Exit-Nodes
  - Angriff über Traffic-Fingerabdruck Analyse
  - Setzt kompromittierten Eintritts-Knoten voraus

➢ Aber: bisher keine generelle IP-Adressen Entschlüsselung!!

Andere Ansätze

- Offene Proxy-Server
- Java Anon Proxy (auch JAP oder JonDonym):
  - Kaskadierende Anonymisierungs-Proxies
  - Explizite Klienten-Konfiguration erforderlich (damit Vergabe von Vertrauen)
- I2P (Invisible Internet Protocol):
  - „Garlic routing“: Gebündelte Nachrichten sollen Traffic-Analyse erschweren
  - Chat, Filesharing und E-Mail Subsysteme
  - Anbindung von anderen Anwendungen: durch I2PTunnel (vergl. mit SOCKS)

Weiterführendes

- Tor (inkl. ARM): torproject.org
- OpenVPN: openvpn.net
- JonDonym: anonymous-proxy-servers.net
- Onion-Site Traffic-Analyse: www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-kwon.pdf
- NSA Tor Präsentation: https://edwardsnowden.com/docs/docs/tor-stinks-presentation.pdf

Zusammenfassung

- OpenVPN:
  - FOSS-Tool der Wahl für VPN (Selbstbau und -Anbieter)
  - OpenVPN vs. binäre Klienten
- Tor:
  - *Das* Anonymisierungs-Netzwerk
  - Bisher keine nachgewiesene Kompromittierung

Diskussion / Fragen

Vielen Dank!

Dr. Christoph Zimmermann

monochromec@gmail.com