Karlsruher Institut für Technologie

Datenschutz und Privatheit in vernetzten Informationssystemen

Kapitel 6: Datenschutz im Internet - Protokollebene

Erik Buchmann (buchmann@kit.edu)

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“

Inhalte und Lernziele dieses Kapitels

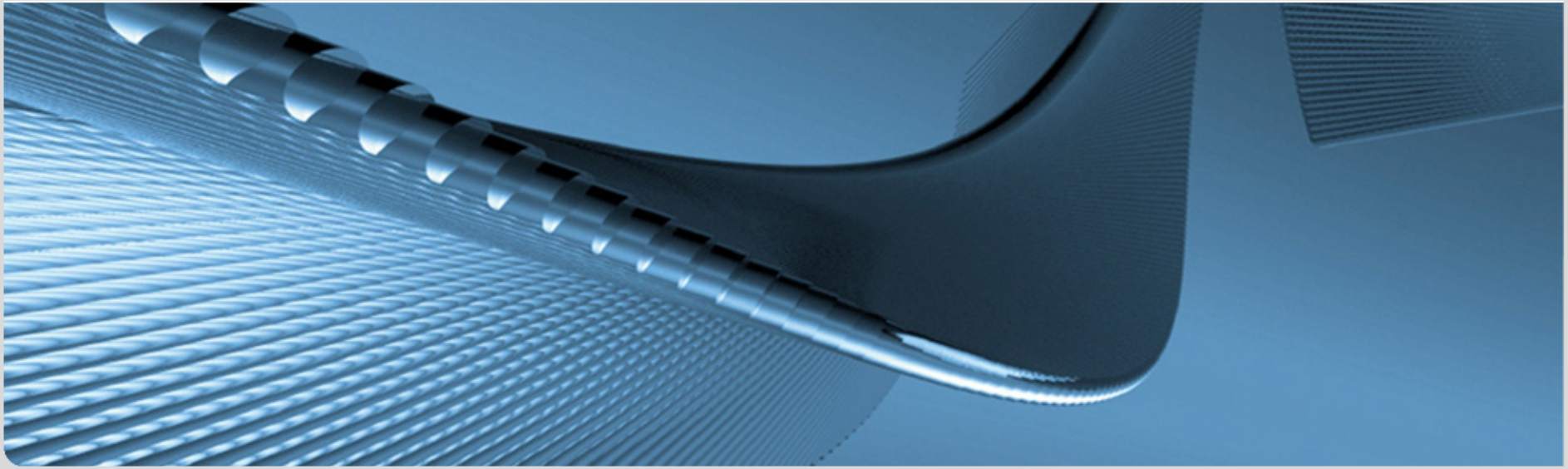
- Einführung
- Datenschutzverfahren auf Protokollebene
  - Mixe nach Chaum
  - JAP
  - TOR
  - Freenet
- Abschluss

- Lernziele
  - Sie können zwischen den Gefahren für den Datenschutz differenzieren, die sich durch die IP-Adresse beim Datenabruf und bei der Datenbereitstellung ergeben.
  - Ihnen ist das Konzept der Mixe im Detail vertraut.
  - Sie können dafür gebräuchliche Schutzverfahren sowie deren Vor- und Nachteile diskutieren.

Karlsruher Institut für Technologie

Einfachste Anonymisierungsverfahren für die IP-Adresse

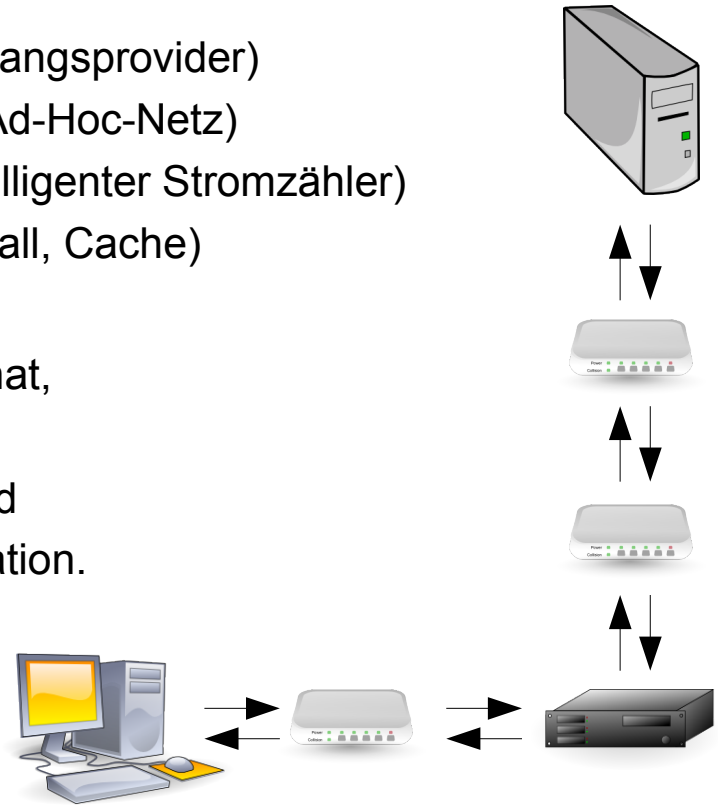
IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Datenschutzproblem IP-Adresse

- Viele verteilte Systeme kommunizieren über das Internet-Protokoll
  - Ethernet, WLAN-Protokolle, Powerline, diverse Funktechnologien

- Im verteilten System kann jeder
  - Einstiegspunkt (Basisstation, Internet-Zugangsprovider)
  - Weiterleiter (Router, Zwischenknoten im Ad-Hoc-Netz)
  - Endpunkt (Webserver, Sensorknoten, intelligenter Stromzähler)
  - zwischengeschaltete Dienst (Proxy, Firewall, Cache)
  - ermitteln,
    - dass eine Kommunikation stattgefunden hat,
    - wer die Kommunikationspartner waren,
    - welcher Art diese Kommunikation war, und
    - den Inhalt (unverschlüsselter) Kommunikation.



Angreifer

- **schwache Angreifer**
  - betreibt Webserver, Router, oder Analysedienst (vgl. Web-Bugs)
  - kann Informationen von wenigen ausgewählten Rechnern zusammenführen
- **starker Angreifer**
  - z.B. Geheimdienst mit starken Ressourcen, Betreiber eines Internet-Backbones
  - kann sehr große Teile der Internet-Kommunikation belauschen

Unterschiedliche Angreifermodelle

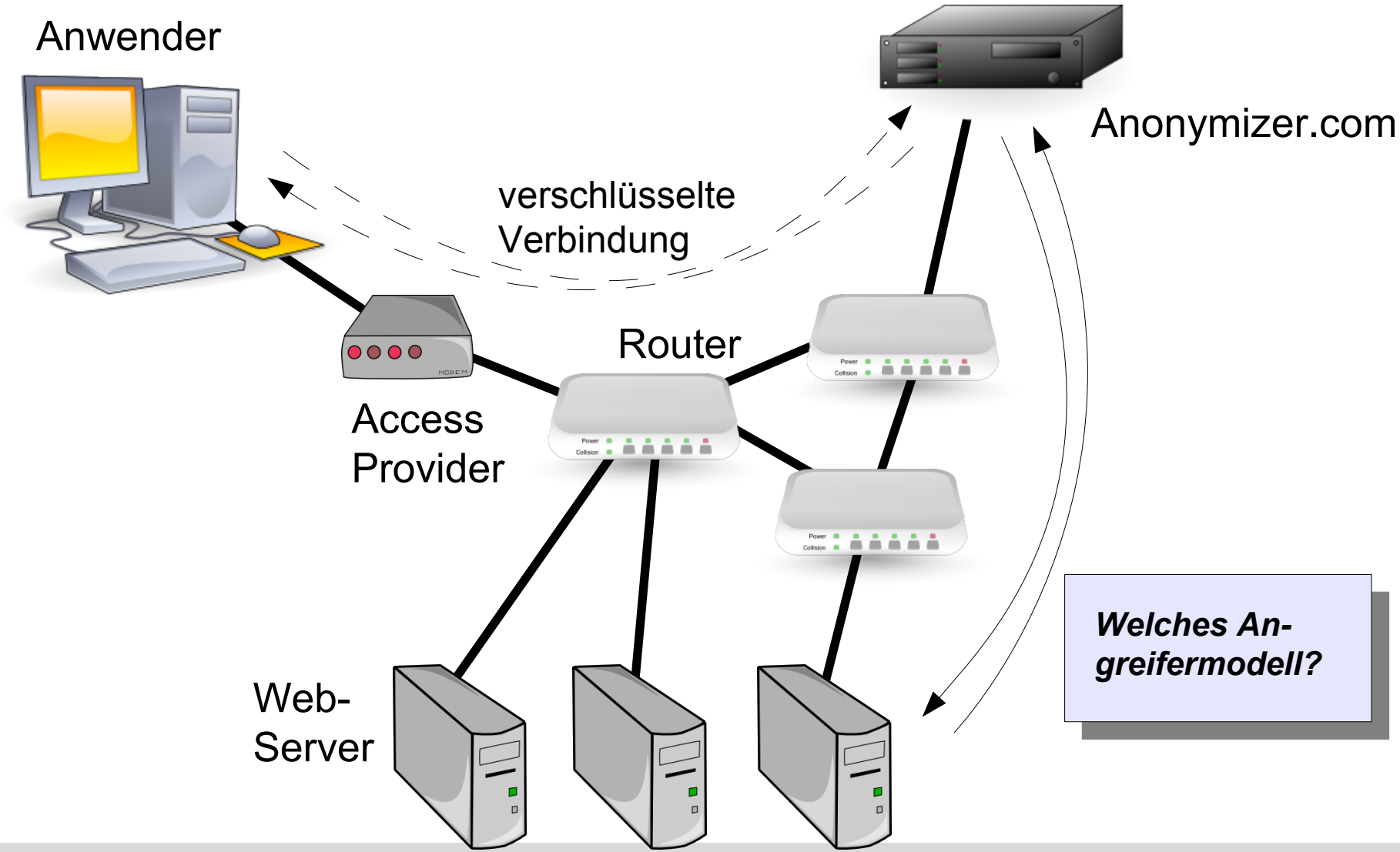
- Senderanonymität
  - Absender einer Nachricht kennt Empfänger,
    Empfänger erfährt nicht die Identität des Senders
- Empfängeranonymität
  - Absender einer Nachricht kennt den Empfänger nicht,
    Nachricht wird über einen anonymen Briefkasten weitergeleitet
- Abstreitbarkeit, vollständig anonymer Datenaustausch
  - Speicherort von Daten ebenfalls unbekannt,
    d.h. kein anonymer Briefkasten mehr vorhanden, über den sich Kommunikationsvorgänge erkennen ließen
  - Nachrichten werden über kryptographische Hash-Signaturen adressiert

Anonymität bei schwachen Angreifern

- Trusted third Party, z.B. http://www.anonymizer.com
  - leitet eingehende Verbindungen unter der eigenen IP-Adresse weiter
  - löscht Cookies, identifizierende Attribute, Referer etc.

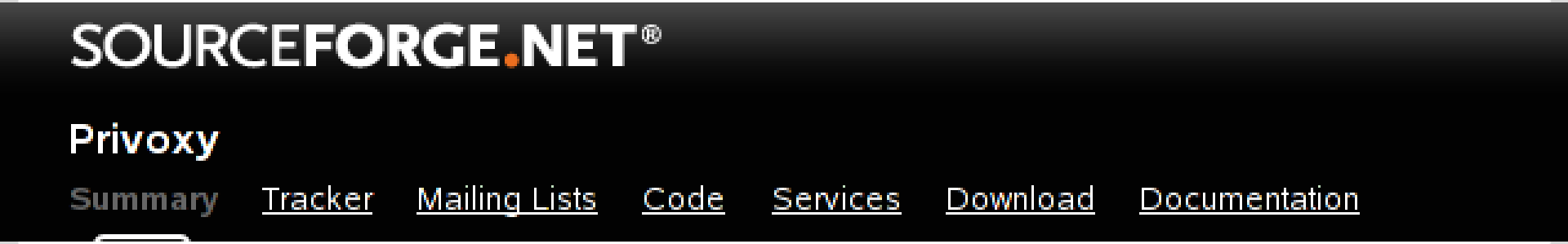
Weitere: iPredator, Relakks, PRQ, Ivacy, Linkideo Perfect Privacy, Surfonym, SwissVPN, Cinipac, FlashVPN, TorrentPrivacy, TrilightZone, StrongVPN, VPNUK, HotSpotShield...

Anonymizer.com



Privoxy

- Privacy-Proxy zum selber-aufsetzen
  - d.h., benötigt einen Rechner, um darauf die Software zu installieren
- Leistet dasselbe wie Anonymizer.com
  - Cookie-Management
  - säubern von HTTP-Headern (Referer, Browser-Informationen, etc.)
  - Filter zum entfernen von Web-Bugs, Scripten
- Open Source, http://www.privoxy.org/



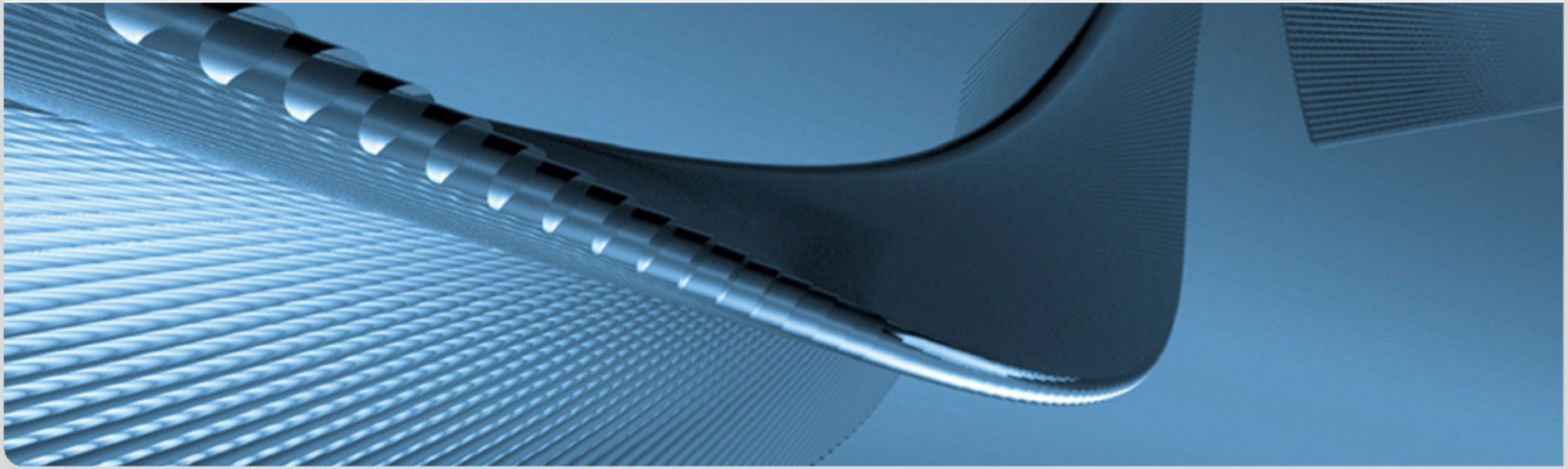
Anonymität bei starken Angreifern

- Es werden gebraucht:
  - starke **Public-Key-Verschlüsselung**
  - möglichst **viele unabhängige Rechner**, welche Nachrichten für den Nutzer weiterleiten
- Idee: Mix-Kaskaden
  - Nutzer verbinden sich nicht direkt mit dem Server, sondern über mehrere Zwischenstationen
  - Kommunikation zwischen zwei Stationen erfolgt verschlüsselt
  - Anfragen eines Nutzers werden zwischen denen anderer Nutzer versteckt
- Implementierungen: JAP, TOR, Freenet
  - Sender-, Empfängeranonymität sowie Abstreitbarkeit möglich

Karlsruher Institut für Technologie

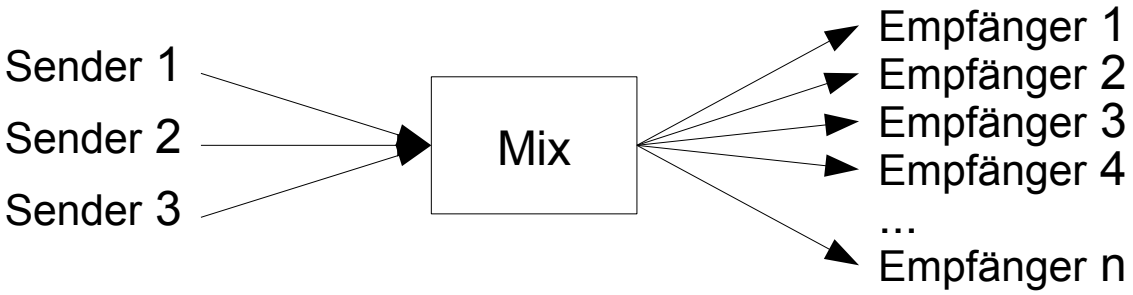
Mixe und Mix-Kaskaden

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Mixe

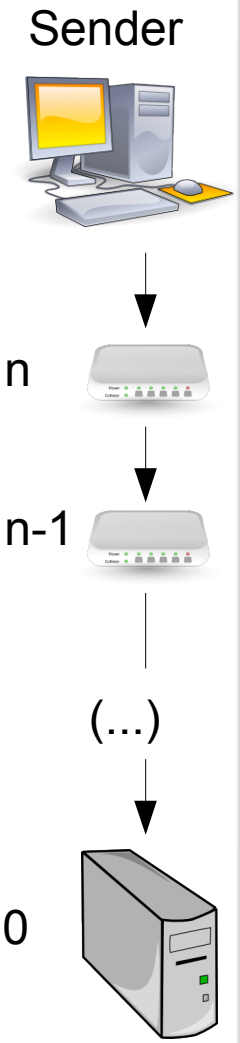
- D. Chaum: *Untraceable Electronic Mail*, Comm. ACM'81
- Mix vermittelt zwischen vielen Sendern und Empfängern



- Jede eingehende Nachricht
  - entschlüsseln (*Public-Key-Kryptografie*)
  - kryptografisch transformieren (*Eingabe-Nachricht ≠ Ausgabenachricht*)
  - Umsortieren, in zufälliger Reihenfolge (*nicht Eingangsreihenfolge!*)
  - weiterleiten zum Ziel (oder weiterem Mix)

Verschlüsselung in Mix-Kaskaden

- Symbole
  - K öffentlicher Schüssel
  - R Zufallszeichenfolge
  - A Adresse der nächsten Station
- Arbeitsweise eines Mix
  - Mix n erhält
    $K_n(R_n, K_{n-1}(R_{n-1}, K_{n-2}(R_{n-2} \ldots Msg, A_0 \ldots A_{n-2}), A_{n-1}), A_n)$
  - mit K_n verschlüsselte Nachricht auspacken,
    R_n löschen, Adresse A_n lesen
  - sendet $K_{n-1}(R_{n-1}, K_{n-2}(R_{n-2} \ldots Msg, A_0 \ldots A_{n-2}), A_{n-1})$ an A_n
  - Letzter Mix sendet Msg unverschlüsselt an Adressaten A_0
    (kann natürlich auch verschlüsselt erfolgen)



Entschlüsselung in Mix-Kaskaden

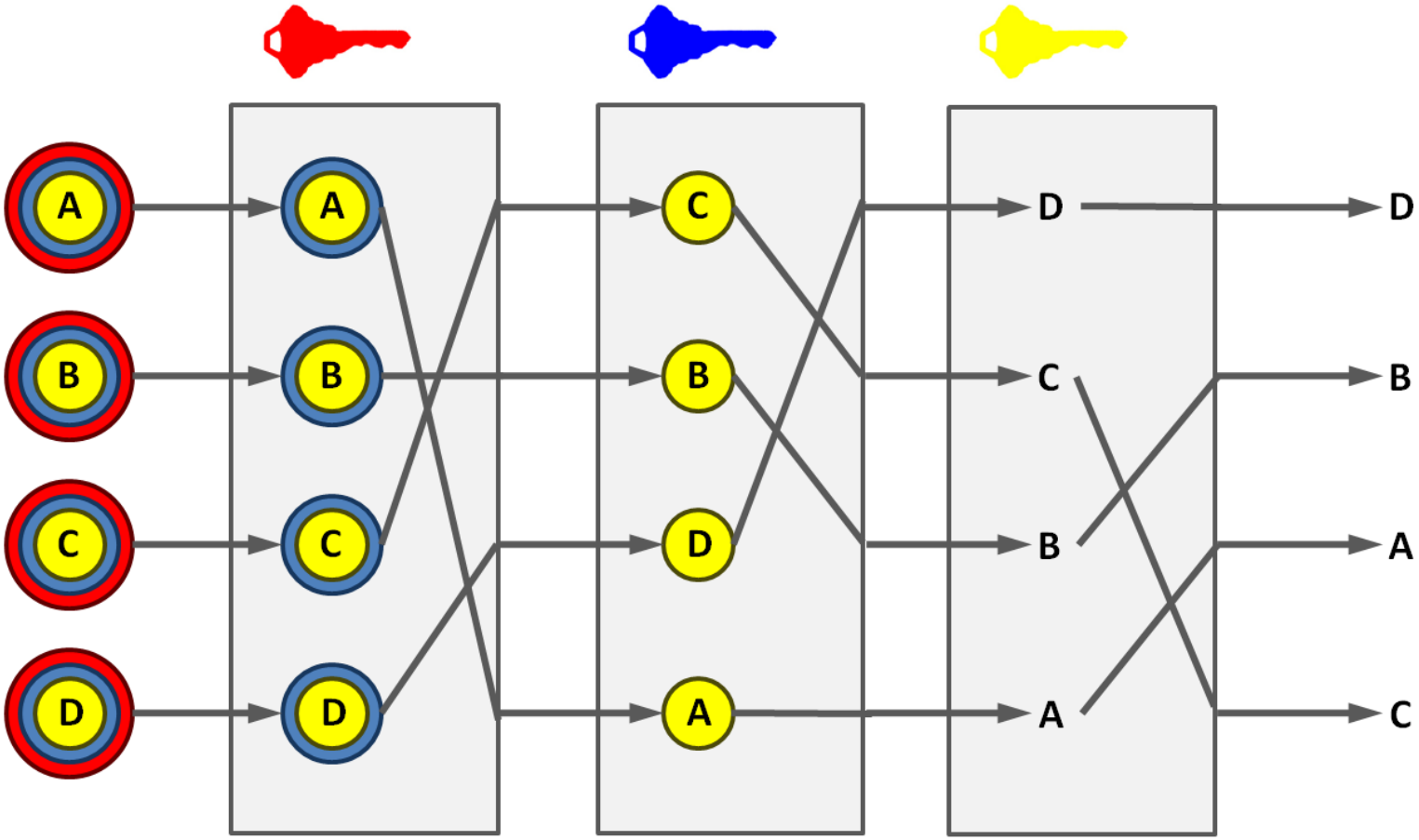
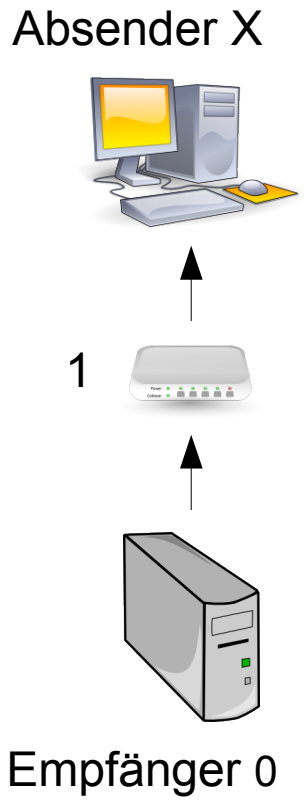


Bild: Wikimedia CC

Was ist mit der Antwort?

- Empfänger soll nicht den Absender feststellen, muss aber trotzdem antworten können

- Prinzip: Adresse für Antwort verschlüsselt mitsenden, R dient jeweils als Schlüssel für die Antwort
  - Absender X, Einmalschlüssel K_X, Empfänger 0
  - Antwortadresse $K_1(R_1, A_X)$, K_X wird zusammen mit ursprünglicher Msg mitgesendet
    - Mix 0 kann Adresse A_X nicht entschlüsseln, sondern nur der letzte Mix in der Kaskade
  - Mix 1 erhält von Mix 0 Antwort $K_1(R_1, A_X)$, $K_X(R_0, Msg)$ und sendet $R_1(K_X(R_0, Msg))$ an A_X
  - Verfahren lässt sich ebenfalls kaskadieren



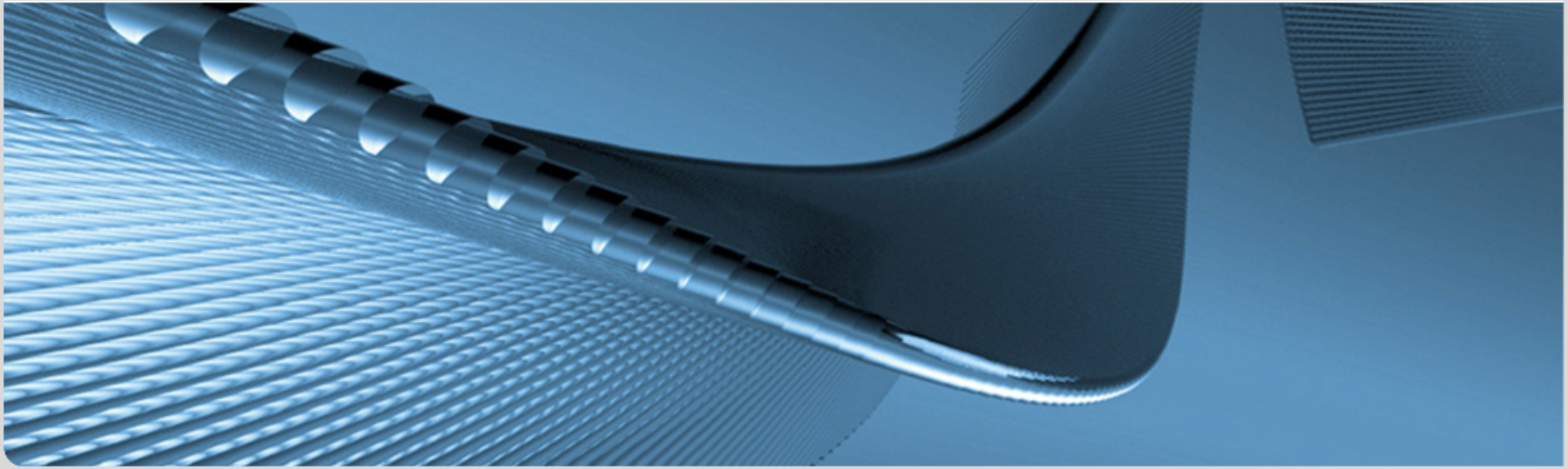
Eigenschaften der Mixe nach Chaum

- Senderanonymität
  - Absender muss Empfänger kennen
- Nur öffentliche Informationen werden übertragen
  - Absender muss kennen:
    - Empfänger
    - öffentliche Schlüssel $K_0 ... K_n$ aller Mixe
  - Jeder Mix m kennt
    - eigenen privaten Schlüssel K^{-1}_m
- Aber: einige *praktische* Probleme
  - Public-Key-Verschlüsselung ist *langsam*
  - statistische Angriffe über die Zahl der ein- und ausgehenden Pakete je Mix
  - Replay-Angriffe (Wiedereinspielen von gültigen Paketen)

Karlsruher Institut für Technologie

Verbergen der IP-Adresse mit JAP

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



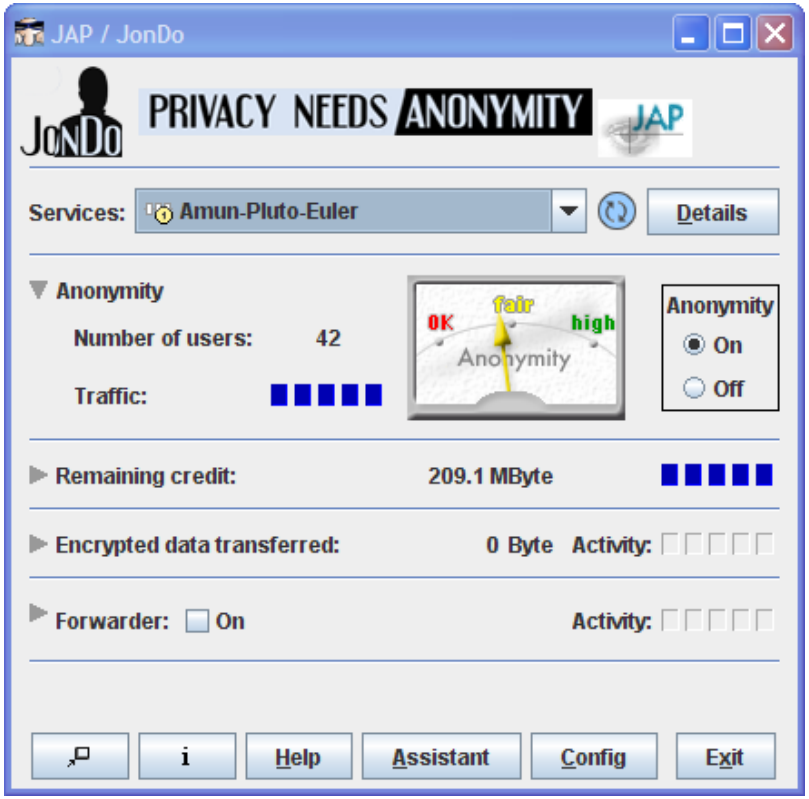
JAP Anon Proxy

- Entwickelt seit 2000 vom AN.ON-Projekt
  - Uni Regensburg, TU Dresden, gefördert mit Mitteln der Deutschen Forschungsgemeinschaft und vom Bundesministerium für Wirtschaft und Technologie
  - http://anon.inf.tu-dresden.de

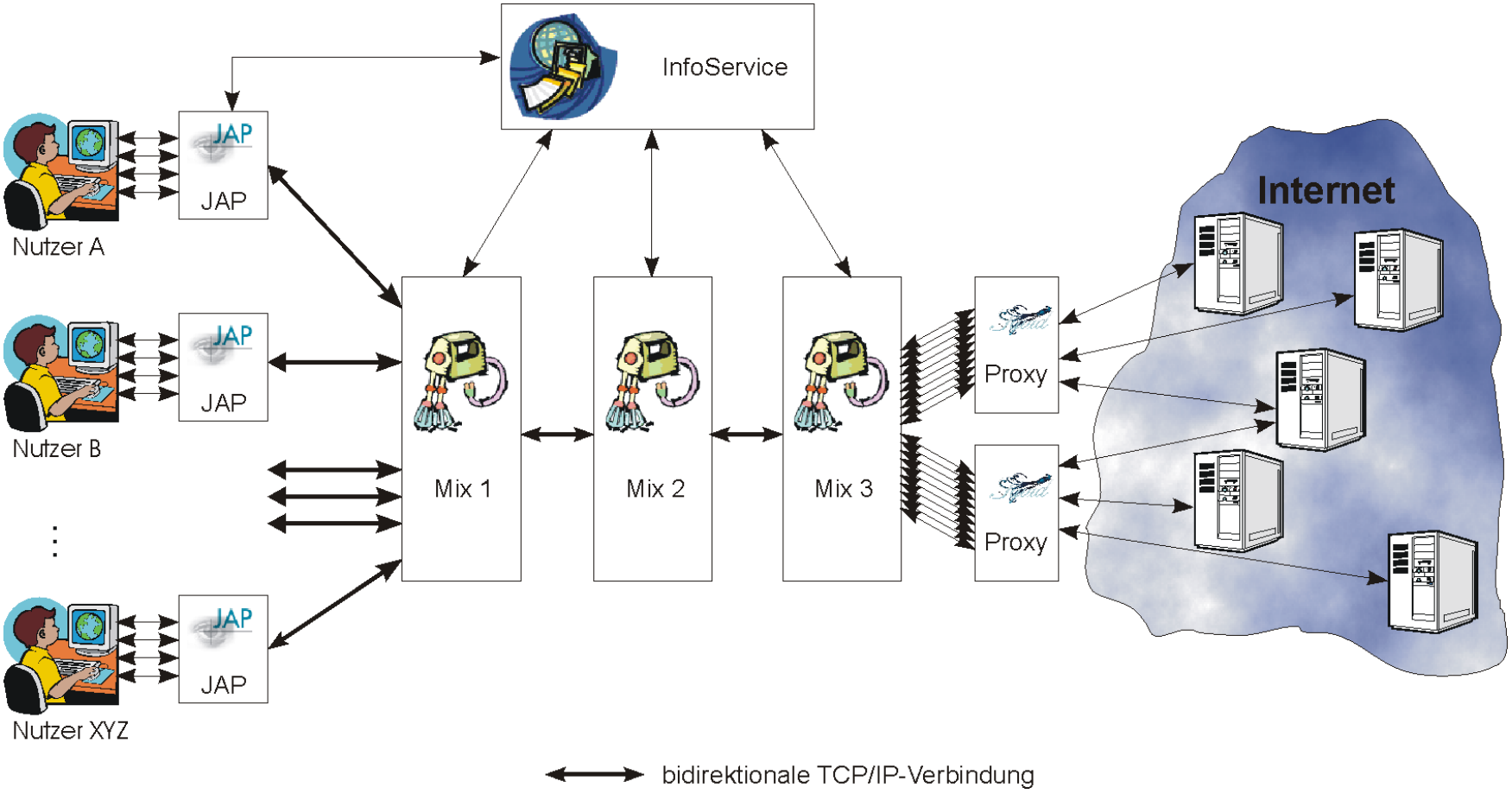
- Ziele:
  - Senderanonymität beim Abruf von Webseiten
    (Anm.: Unterschied zu TOR o.ä., welche die gesamte Internet-Kommunikation verschlüsseln können)
  - Einfache Benutzbarkeit; leichte Installation, Konfiguration und Bedienung
  - Dienstgüte; sowohl bzgl. Netzparametern (Durchsatz, Latenzzeit etc.) als auch Sicherheit (Anonymität)

JAP Anon Proxy

- Im Gegensatz zu Chaum **feste Kanäle** mit **symmetrischer Verschlüsselung**
  → Problem mit der Rücksendeadresse entfällt,
  symm. Verschlüsselung ist effizienter als Public-Key
- grafische Client-Komponente
  - Proxy im Webbrowser
- Zusätzlich: Info-Service informiert über Schlüssel und aktive Mixe



Architektur von JAP

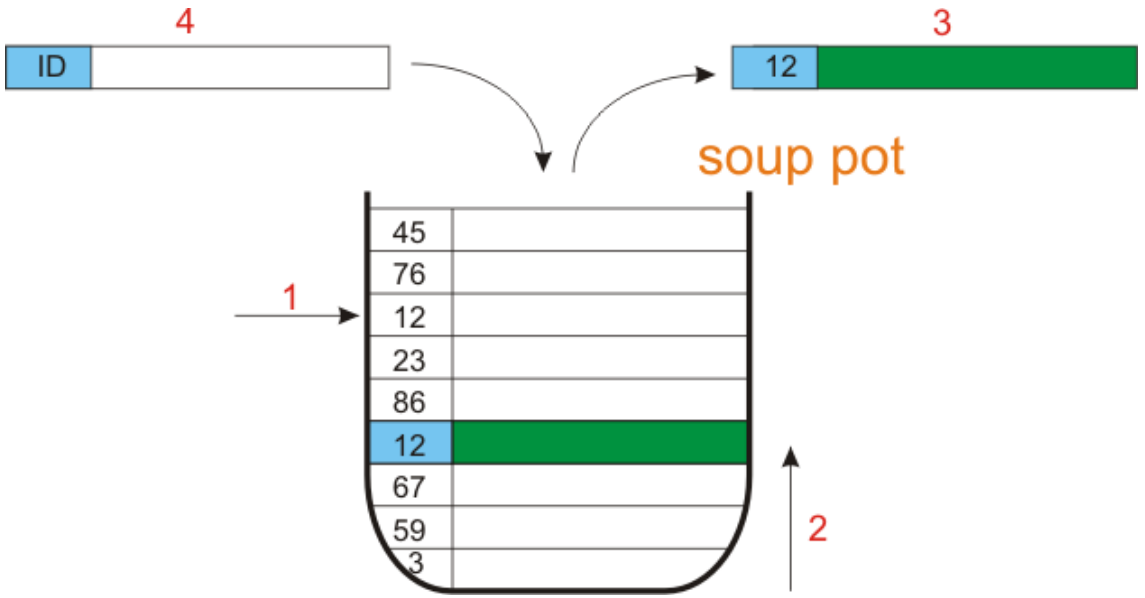


Quelle: S. Köpsell, http://anon.inf.tu-dresden.de

Mix-Kanäle

- Statische Verbindung zwischen zwei Kommunikationsendpunkten über mehrere Mixe hinweg
  - Verbindungsorientierter Vollduplex-Betrieb
    (Gegensatz zu paketorientiert bei Chaum)
  - vom Sender einer Nachricht initiiert
- Sicherheitsfeatures
  - Datenstrom wird auf mehrere MIX-Pakte aufgeteilt, die alle gleich groß sind (auffüllen mit Zufallszahlen)
  - Zufällige Pakete an beliebige Empfänger, um Zuordnung zu verhindern
  - Symmetrische Verschlüsselung; (langsame) Public-Key-Verschlüsselung für Aushandeln der symmetrischen Schlüssel

Umsortieren von Paketen: Mix-Pool

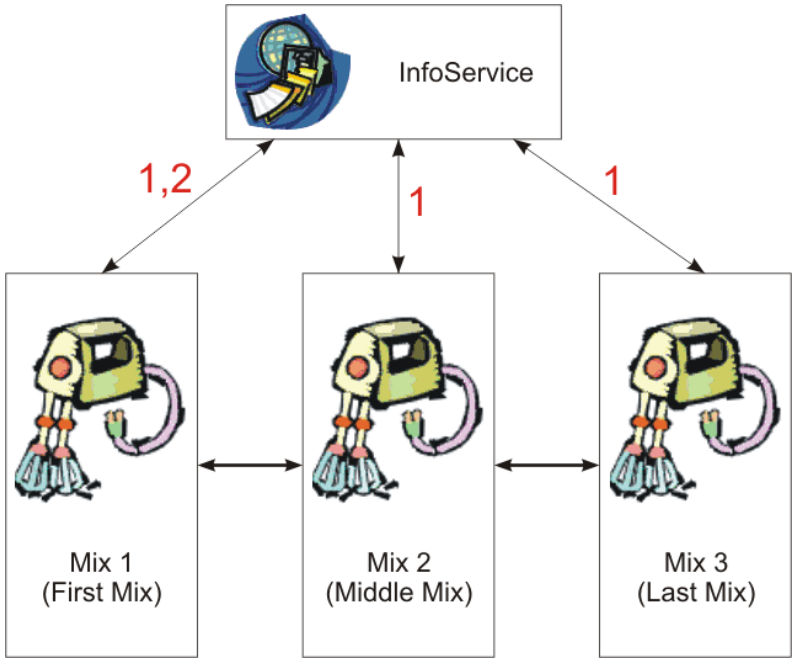


- Eintreffen eines MixPaketes:
  1. Zufällige Auswahl eines MixPaketes (hat *Kanal-ID*)
  2. Suchen nach dem ältesten MixPaket mit *Kanal-ID* im Pool
  3. Ausgabe des MixPaketes
  4. Hinzufügen des empfangenen MixPaketes

Quelle: S. Köpsell, http://anon.inf.tu-dresden.de

Kommunikation mit dem Info-Service (1/2)

1. jeder Mix sendet Informationen über sich alle 10 Minuten an den InfoService (Name, Betreiber, Standort etc.)
2. erster Mix einer Kaskade sendet Statusinformationen (Benutzer, gemixte Pakete) jede Minute an InfoService

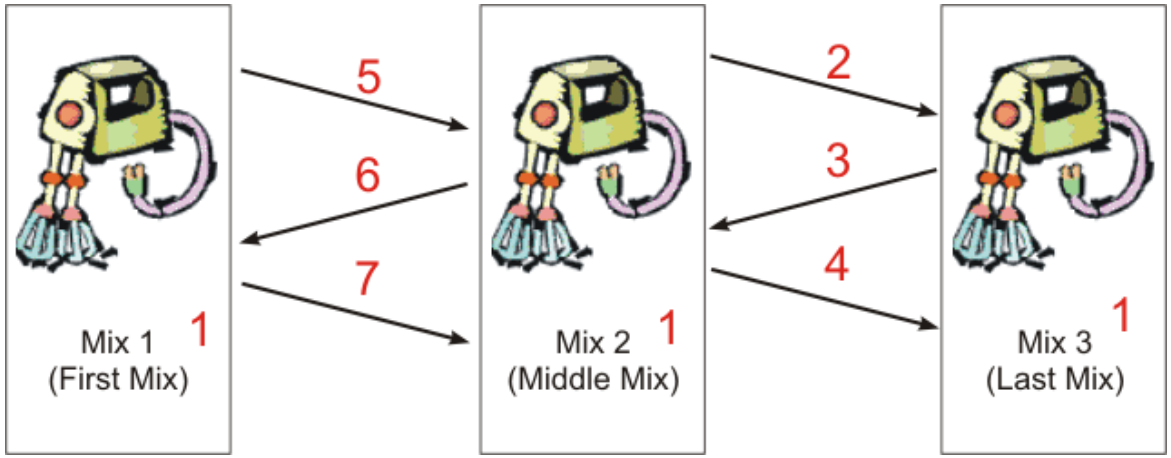


Quelle: S. Köpsell, http://anon.inf.tu-dresden.de

Kommunikation mit dem Info-Service (2/2)

- Infoservice informiert über
  - Status-Daten zum Mix
  - Mix-Betreiber
  - aufgebaute Mix-Kaskaden
- Dient als Verzeichnis:
  - Auslieferung der JAP-Software
  - Verzeichnis der aktiven Mixe
  - Verzeichnis der öffentlichen Schlüssel
- JAP funktioniert generell aber auch ohne
  - jedenfalls sofern der Nutzer selbst Adresse und Schlüssel vom ersten Mix in einer Kaskade kennt

Starten einer Mix-Kaskade



1. alle Mixe werden gestartet
2. Mix 2 verbindet sich zu Mix 3
3. Mix 3 sendet öffentlichen Schlüssel
4. Mix 2 sendet symmetrischen Schlüssel (verschlüsselt und signiert)
5. Mix 1 verbindet sich zu Mix 2
6. Mix 2 sendet seinen öffentlichen Schlüssel (signiert von Mix2) und den von Mix 3 (signiert von Mix 3)
7. Mix 1 sendet symmetrischen Schlüssel (verschlüsselt und signiert)

Quelle: S. Köpsell, http://anon.inf.tu-dresden.de

Verbindungsaufnahme mit einer Kaskade

1. Client etabliert TCP/IP-Verbindung zum ersten Mix einer Kaskade
2. Mix sendet signierte Liste mit je einem Eintrag pro Mix der Kaskade an Client:
   - XML Struktur, jeder Eintrag enthält:
     - öffentlichen RSA Schlüssel des Mixes
     - ID des Mixes
     - Signatur geleistet von Mix
3. Client sendet symmetrischen Schlüssel für Verbindung von JAP ↔ erstem Mix der Kaskade an Mix (verschlüsselt mit öffentlichem Schlüssel des Mix)
4. Verbindung kann benutzt werden

Die JAP-Mixe

Name	Nutzer	Verkehr	Gemixte Pakete
Dresden-Dresden	2424	100 (hoch)	4,222,626,257
Koelsch-Rousseau-Zeuten	75	100 (hoch)	4,820,805
Fondue-Montesquieu-Uranus	93	100 (hoch)	30,503,809
Koala-SpeedPartner-Titan	70	68 (hoch)	1,303,332
FreeBeer-Bolzano	400	100 (hoch)	242,911,019
Opossum-Grolsch-Transformer	192	100 (hoch)	212,382,921
HE-Zeitwort	296	51 (normal)	1,448,220

Stand: 13.05.2013, 14:00 Uhr

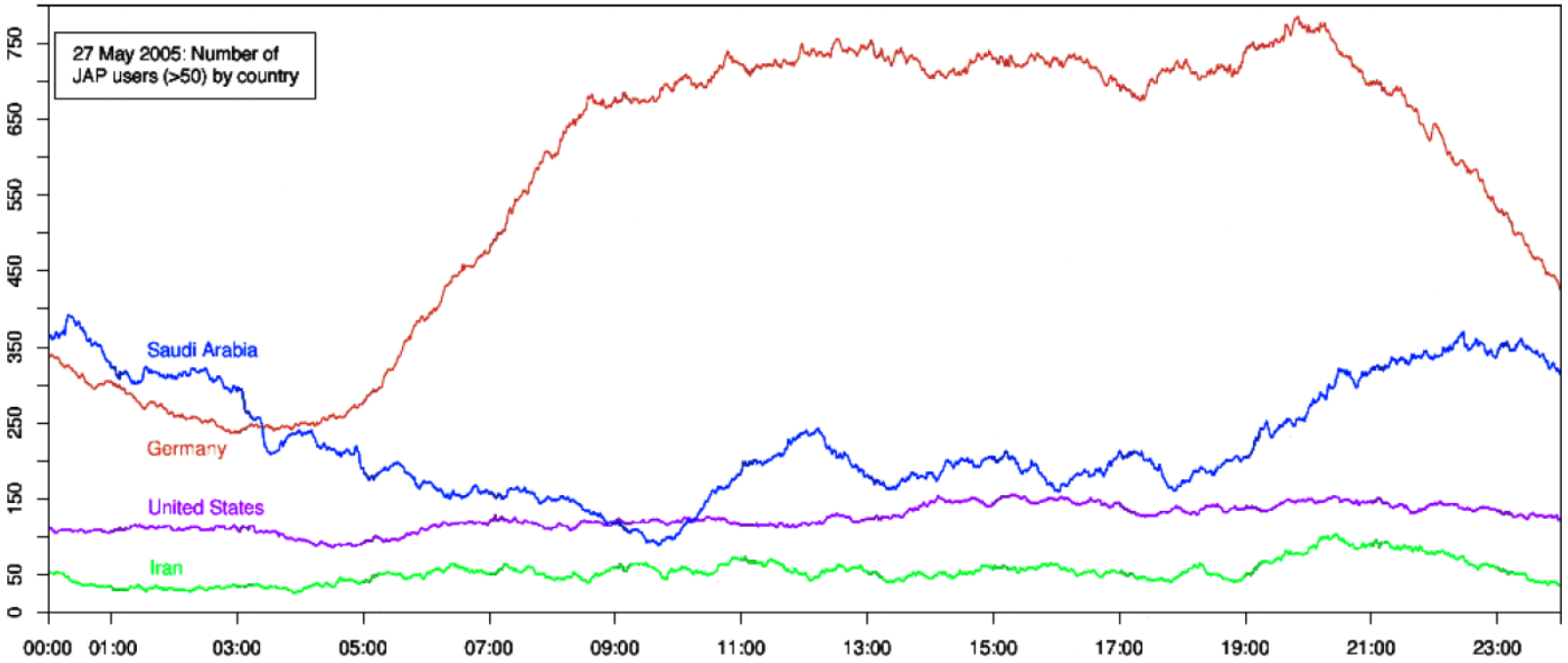
Mix-Betreiber

- Mix-Betreiber gibt eine Selbstverpflichtung ab
  - keine Logfiles über die in JAP transportierten Verbindungen speichern (Anm.: aber Logfiles für eigene Verbindungen, vgl. Vorratsdatenspeicherung)
  - kein Datenaustausch mit anderen Mix-Betreibern
- Verknüpfung von aufgezeichneten Daten aller Mixe unmöglich
  - Mixe generieren für jede neue Kaskade neue Schlüssel, die nicht gespeichert werden (solange Software nicht manipuliert)
  - Schlüssel aus Hauptspeicher gelöscht sobald Kaskade geschlossen
- Solange nicht alle Betreiber einer Mix-Kaskade miteinander kooperieren, ist Anonymität sichergestellt
  - jedenfalls wenn der Benutzer keine Fehler macht

Wer sind die Nutzer von JAP?

- Europa 60%
  Asien 27%
  USA 12%
  Rest 1%

Quelle: http://anon.inf.tu-dresden.de



Und wie verhalten sie sich?

- Anfragen von Strafverfolgern und Privatpersonen zwecks Rückverfolgung einer IP-Adresse

Jahr	Anfragen gesamt	Anfragen von Strafverfolgungs-behörden	Anfragen von Privatpersonen	
2001	13	7	6	Projektstart
2002	40	15	25	
2003	58	21	37	
2004	61	38	23	
2005	42	27	15	
2006	7	4	3	Bis 31. März 2006

Quelle: https://www.datenschutzzentrum.de/projekte/anon

Vorratsdatenspeicherung und JAP

- 01.01.2009: Vorratsdatenspeicherung (§113a TKG)
  - Speicherung von Verkehrsdaten (nicht: Inhalte)
- Umgesetzt in JAP
  - der erste Mix speichert IP-Adresse, Zeit, Kanalnummer zum nächsten Mix
  - mittlere Mixe speichern eingehende und ausgehende Kanalnummern, Zeit
  - ausgehende Mixe speichern die eingehende Kanalnummer, Zeit, Portadresse im Internet
- Nicht gespeichert: URL, IP des kontaktierten Servers

  → *Behörden müssten Daten von Webserver und allen Mixen einer Kaskade kombinieren; es genügt ein Mix im Ausland*

Anmerkung: Bundesverfassungsgericht erklärte März'10 Vorschriften zur VDS für verfassungswidrig, daher im Moment keine VDS.

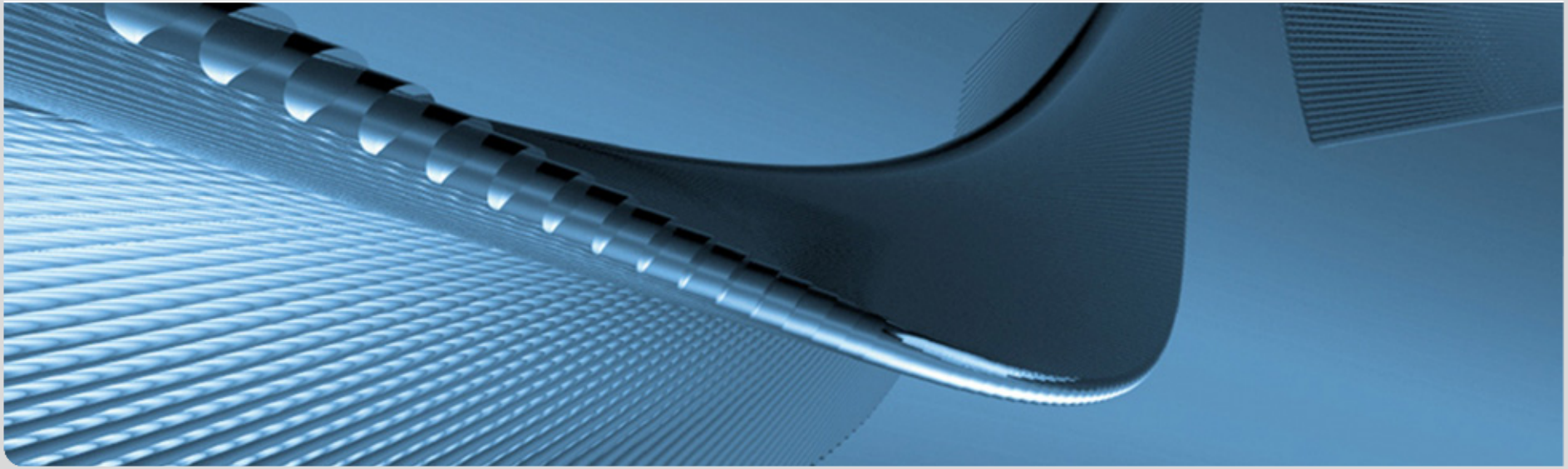
Grenzen des Ansatzes

- Verschleiert nur die IP-Adresse
  - Cookies, Identifizierung durch Quasi-Identifier in Suchmaschinen- oder Formulardaten möglich
    - Aufmerksamkeit, Kenntnisse vom Nutzer gefordert
- Statische Kaskaden
  - Wenn Angreifer errät, welche Kaskade eine Zielperson nutzt, genügt der Angriff von 3 Rechnern
- Beschränkt auf Webseiten-Zugriff
  - allerdings eher aus Performanz-Gründen;
    kein Problem des Konzepts
- Performanz, Verfügbarkeit
  - Mix-Server einer Kaskade sind Single Point of Failure
    und Bottleneck für viele Nutzer

Karlsruher Institut für Technologie

Verbergen der IP-Adresse mit TOR

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Tor

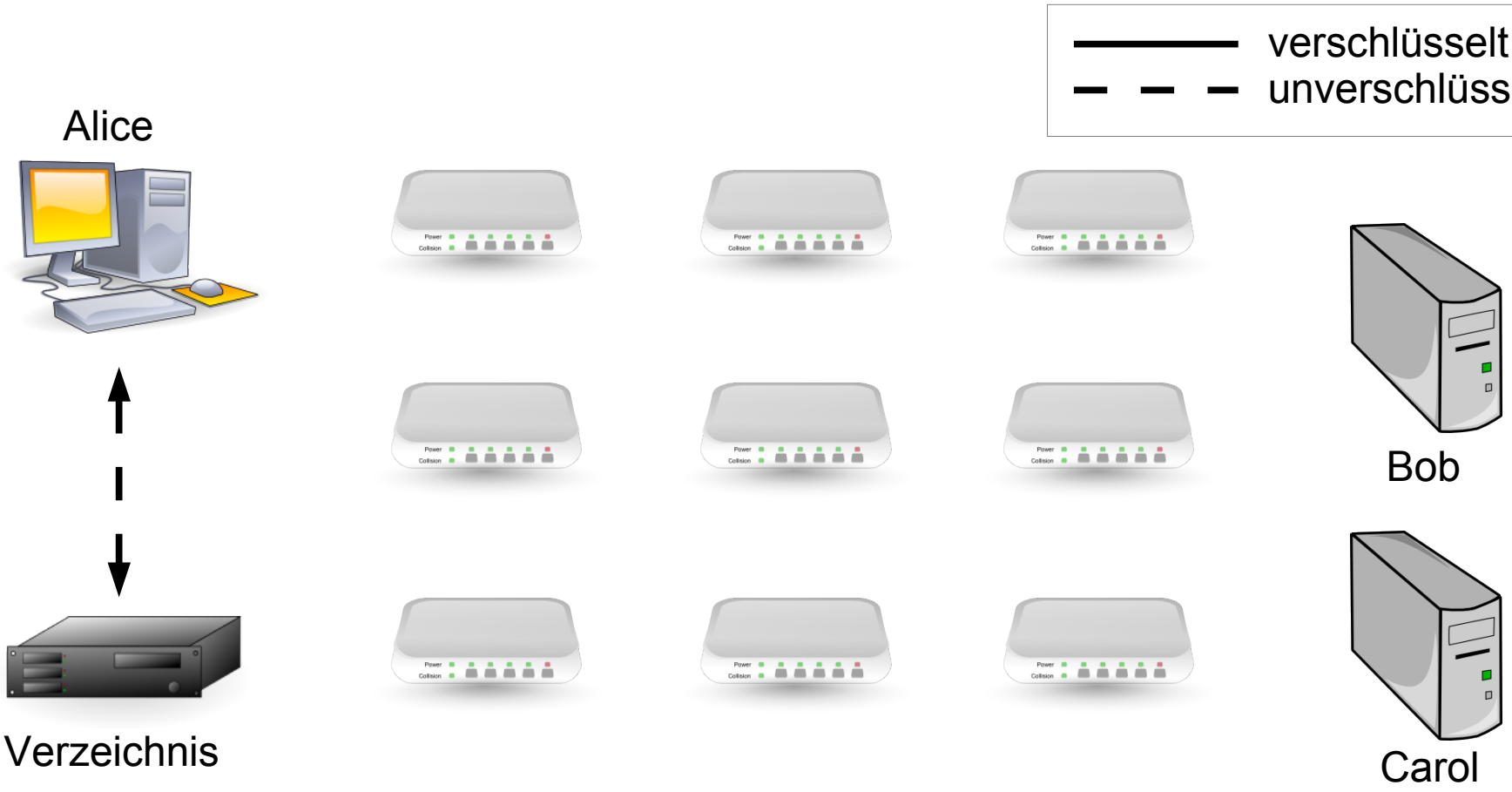
- Projekt der Electronic Frontier Foundation

- Merkmale von Tor
  - **Sender-Anonymität**
    - Onion-Routing (nichts anderes als das Mix-Modell)
  - **Empfänger-Anonymität**
    - Hidden Services: Anonyme Serveradressen
  - Unterstützt beliebige TCP-basierte Protokolle
    - http, P2P-Protokolle oder ICQ
  - Zufällige Verbindungen für jede Netzwerkverbindung
    - d.h. Kanal wird neu gewählt, sobald Server die Verbindung schließt.
  - Persistente Verbindungen
    - Mix-Kaskade dient zum Aufbau eines symmetrisch verschlüsselten Kanals

Roger Dingledine et al.: Tor: The Second-Generation Onion Router, Proceedings of the SSYM 2004

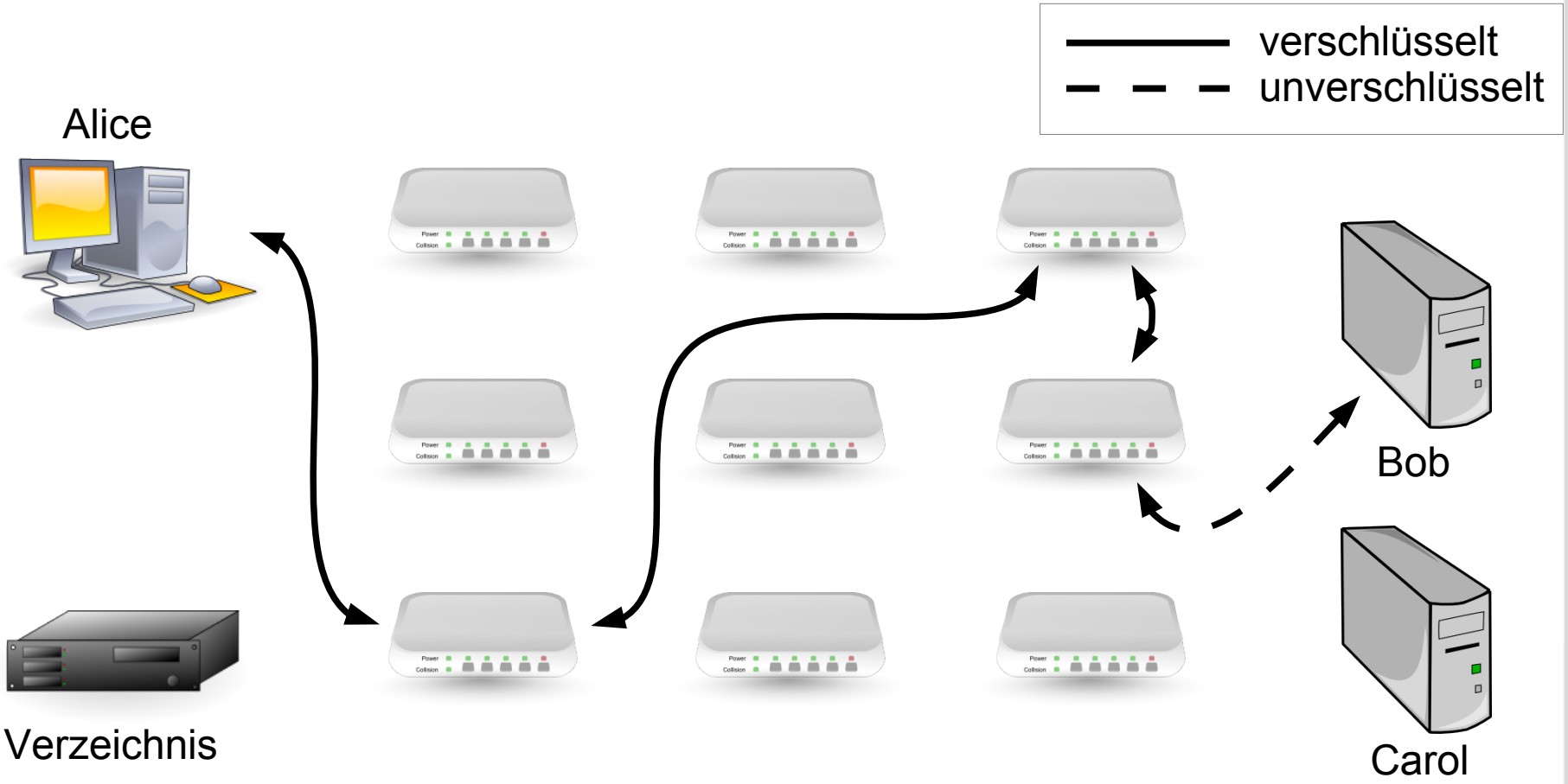
Funktionsweise von Tor (1/3)

- Alice erhält Liste von Torknoten vom Verzeichnisserver



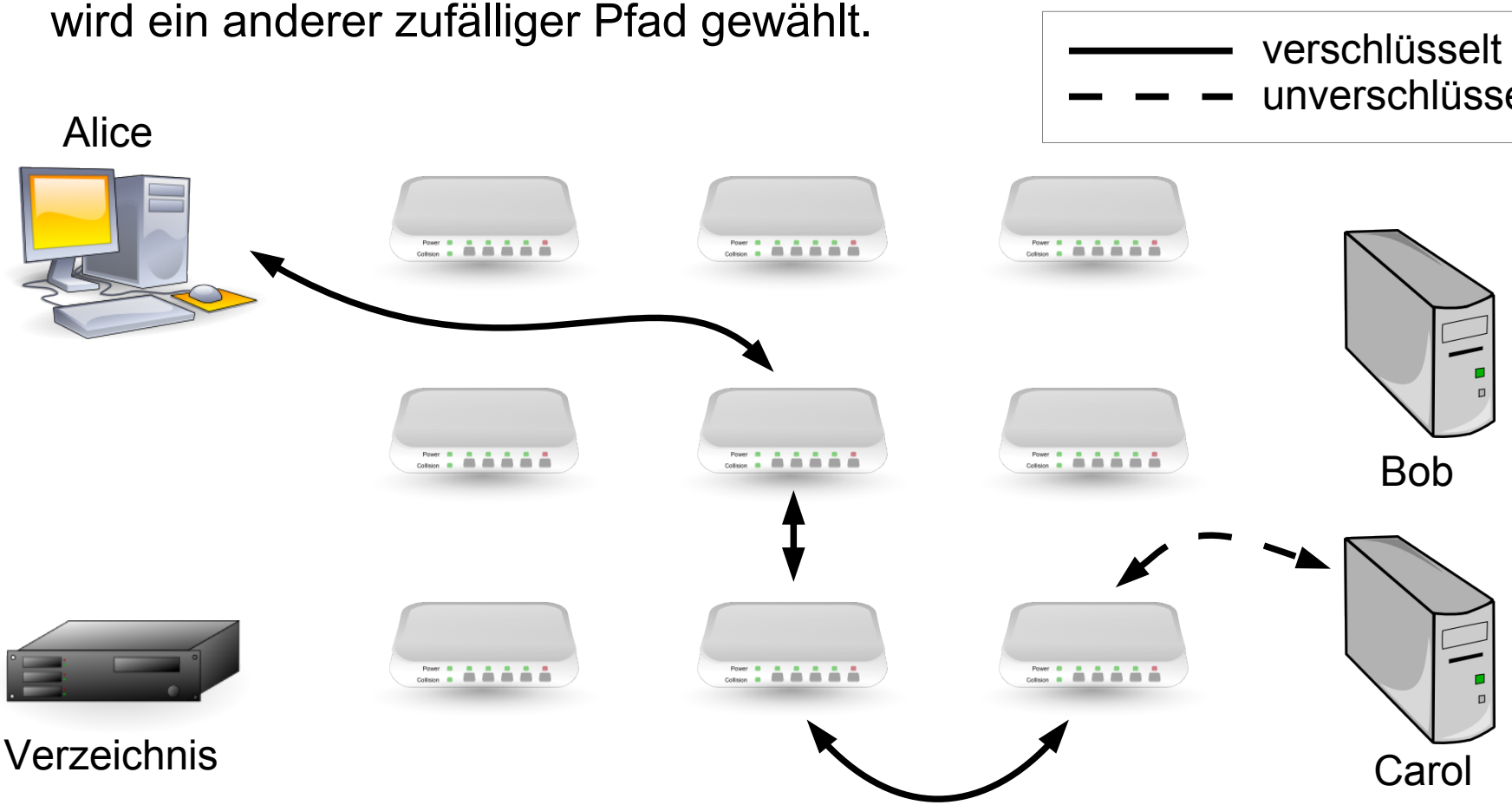
Funktionsweise von Tor (2/3)

- Alice wählt zufälligen Pfad zum Zielrechner



Funktionsweise von Tor (3/3)

- Wenn Alice auf einen anderen Rechner zugreifen will, wird ein anderer zufälliger Pfad gewählt.



Hidden Services

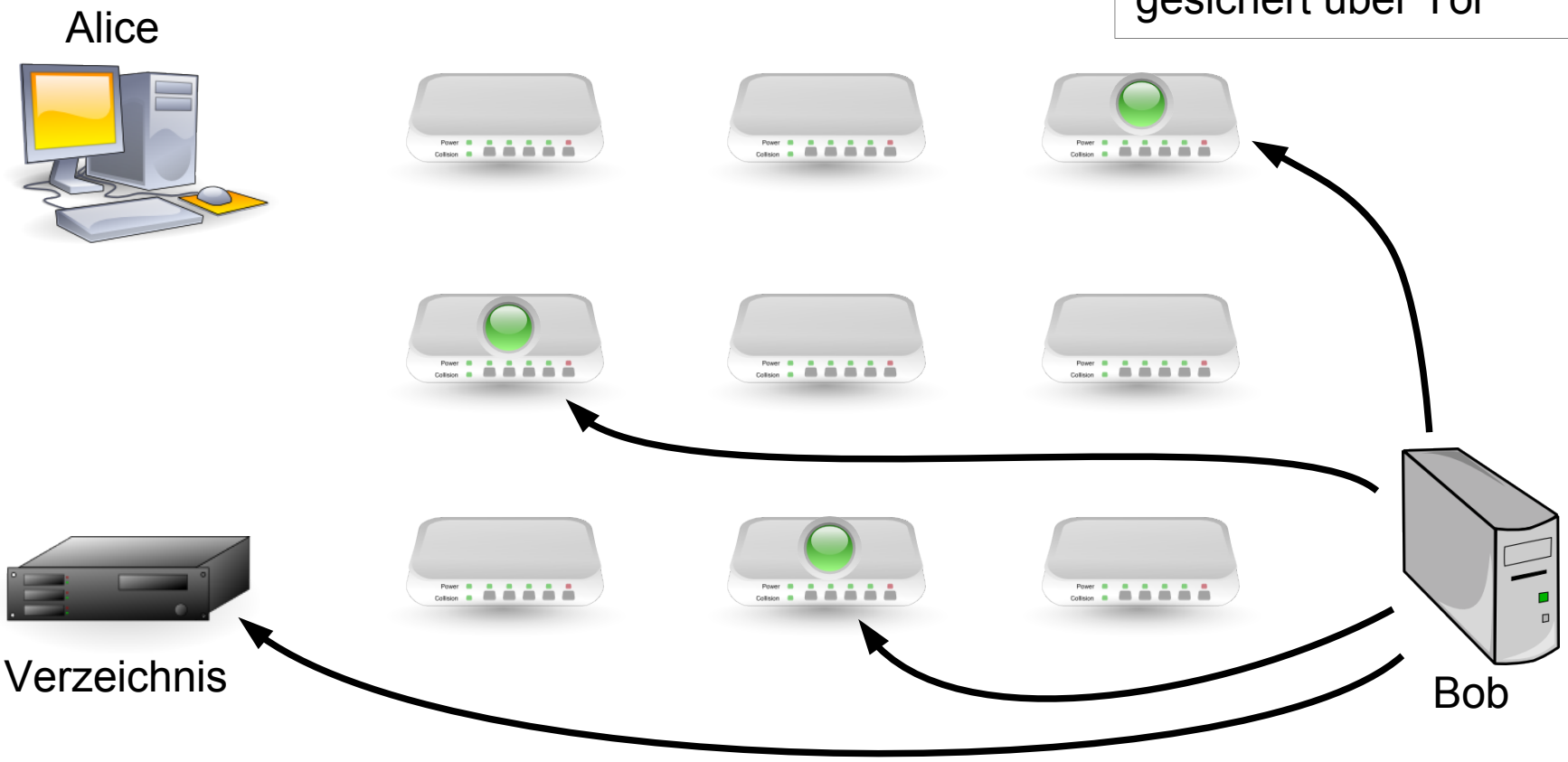
- Ziel: Empfänger-Anonymität
  - Dienste anbieten, ohne die eigene Identität preiszugeben
  - herkömmliches WWW: whois-Anfrage verrät Betreiber

- Bestandteile
  - **Introduction Points** als Verbindung zum Server
  - **Descriptor** auf Verzeichnisserver, der Public Keys und Introduction Points enthält
    - Anonymer Dienst wird über einen eigens generierten **Public Key** identifiziert, der sonst nirgends gebraucht wird
  - **Rendezvous-Knoten** zur anonymen Verbindungsaufnahme

Tor Hidden Services (1/4)

- Bob wählt zufällige Introduction Points aus, und teilt die dem Verzeichnisserver mit

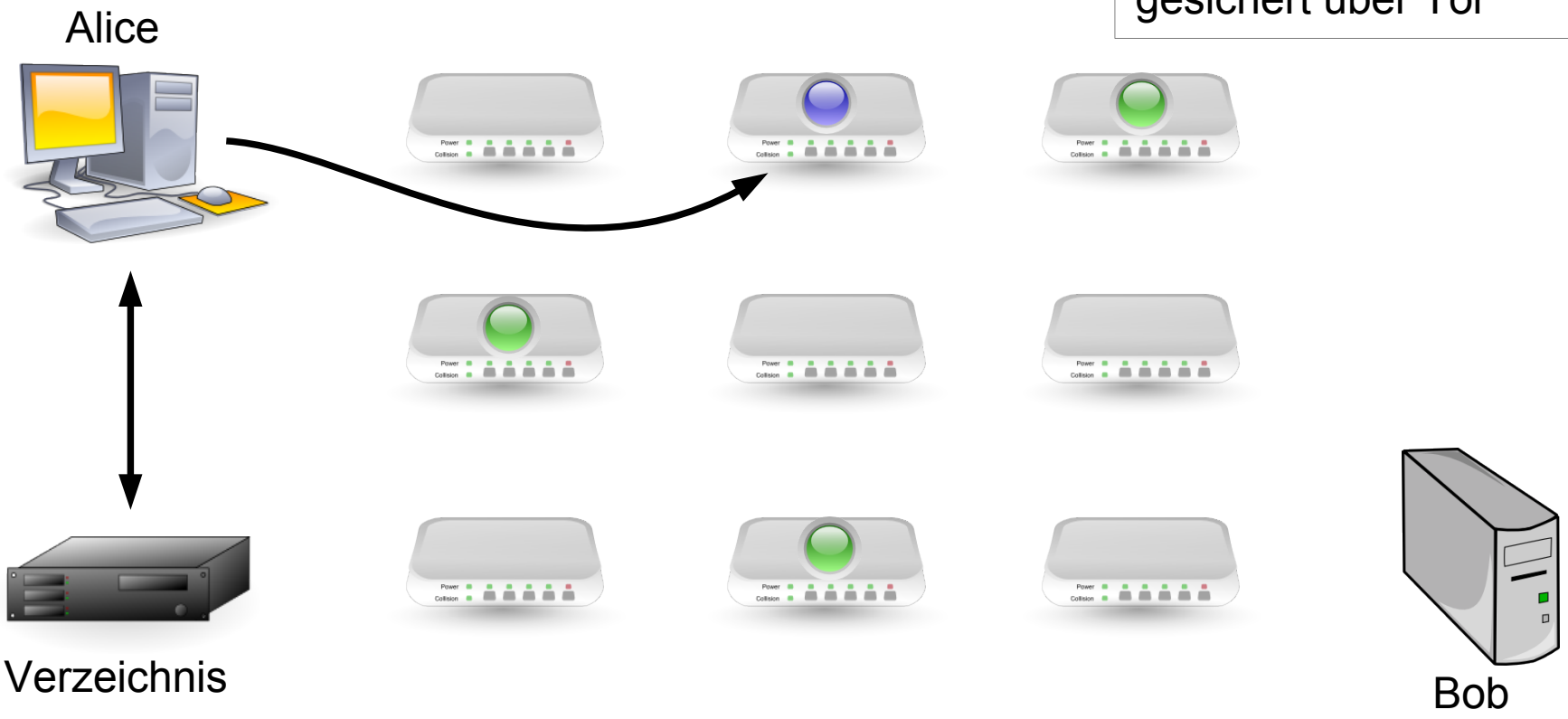
Anm.: alle Verbindungen gesichert über Tor



Tor Hidden Services (2/4)

- Alice fragt Verzeichnisserver nach den Introduction Points von *bob.onion* und legt Rendezvous-Knoten fest

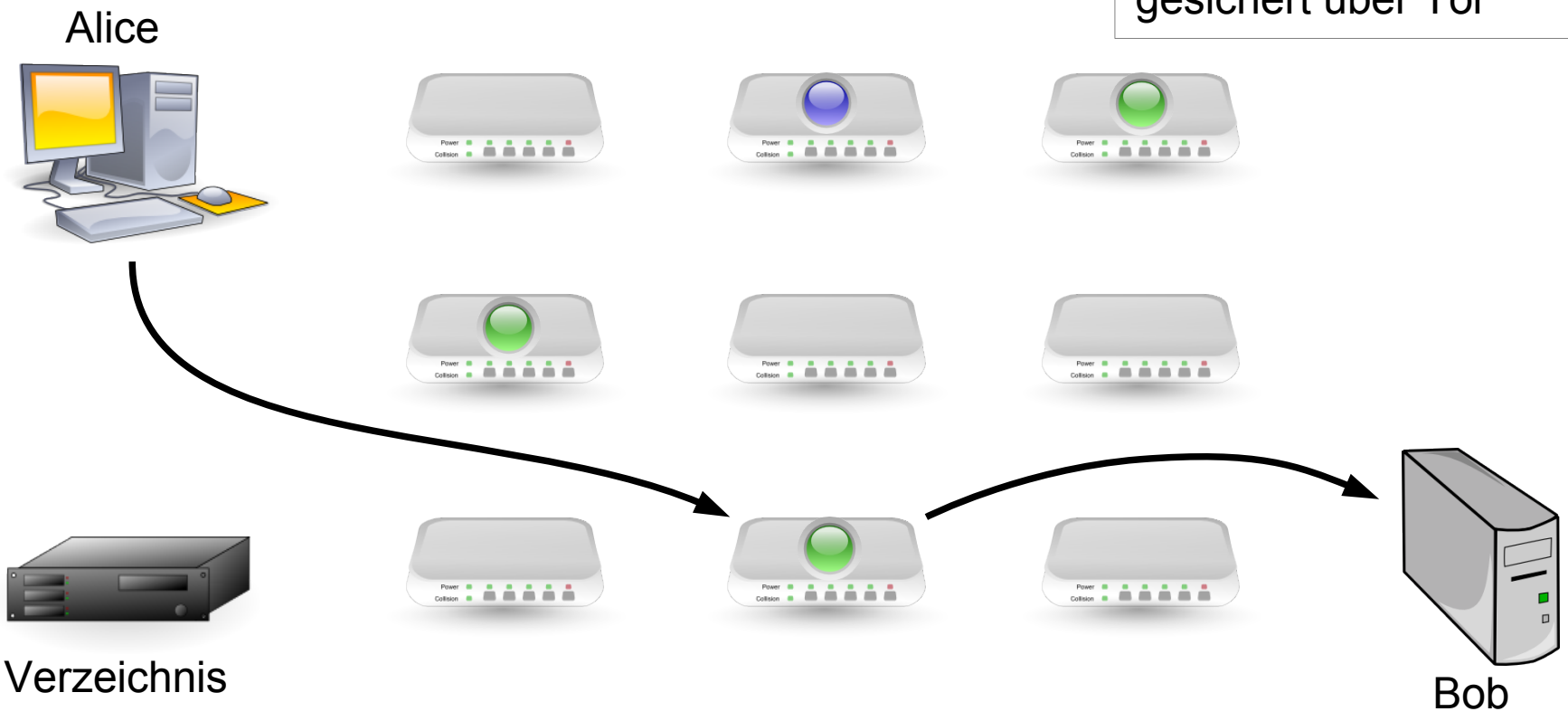
Anm.: alle Verbindungen gesichert über Tor



Tor Hidden Services (3/4)

- Alice schickt Bob über Introduction-Point ein One-Time-Secret und ihren Rendezvous-Knoten

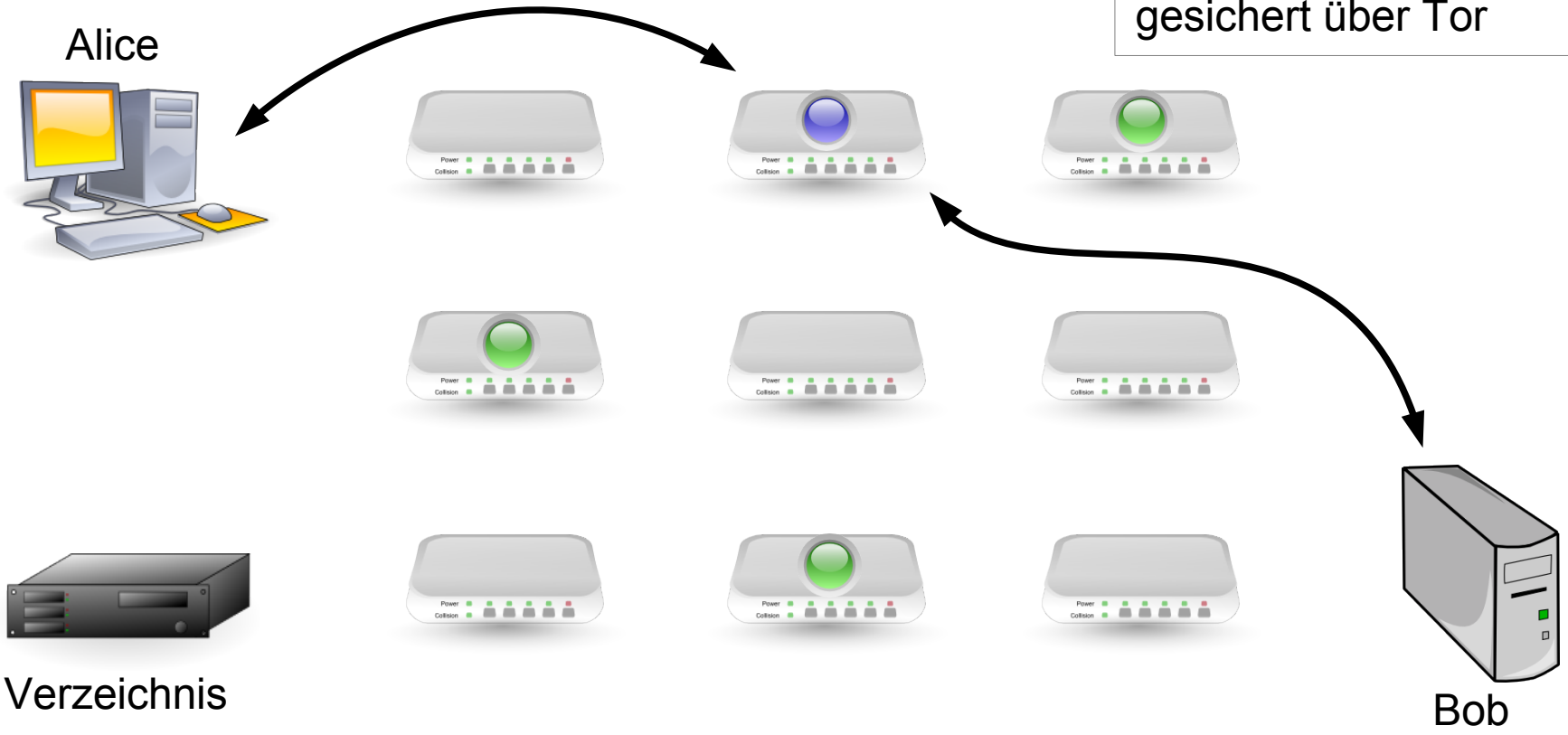
Anm.: alle Verbindungen gesichert über Tor



Tor Hidden Services (4/4)

- Bob antwortet mit dem One-Time-Secret über den Rendezvous-Knoten
- für beide anonyme Verbindung ist hergestellt

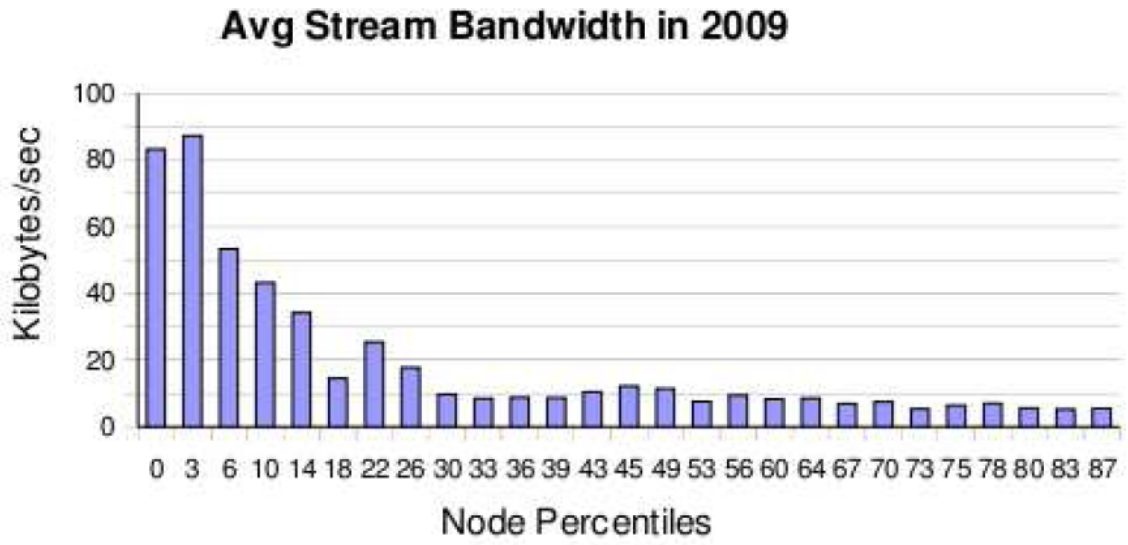
Anm.: alle Verbindungen gesichert über Tor



Warum keine Kommunikation über Introduction Points?

Zahlen zu Tor

- Zahl der Knoten (2007, http://heise.de)
  - ca. 700 Tor-Knoten, welche die Verschlüsselung und den Weitertransport übernehmen
  - ca. 250 Exit-Knoten mit mehr als 20kB/s, die Daten ins Internet weiterleiten
- Performanz (2009)



Mike Perry: TorFlow: Tor Network Analysis, http://fscked.org/projects/torflow, 2009

Grenzen des Ansatzes

- Datenspuren auf verschiedenen 'logischen Ebenen'
  - IP-Adresse ist nicht alles
  - Beispiel Internet
    - Web-Bugs, Cookies, Scripte etc. können Nutzer identifizieren
- einige praktische Probleme
  - Endpoints-Angriff
  - Route Selection-Angriff
  - Selective Disruption-Angriff
  - Sidechannel-Angriffe wie Clock Skew
  - Performanz

Endpoints-Angriff

- Angreifer kontrolliert den ersten und letzten Knoten einer Verbindung
  - Statistischer Angriff über Paketzahl und zeitliche Abfolge

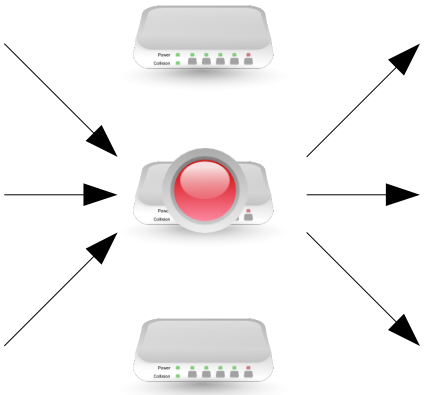
  - Angreifer kontrolliert x von N Knoten, v Verbindungen:
    Wahrscheinlichkeit p, beide Endpunkte zu kontrollieren ist
    $p = v * (x/n)^2$
    für große v → 100%

  - Abhilfe: **Entry Guards**
    - Auswahl von maximal 3 Tor-Knoten als Entry Guards
      - Auswahl nach freien Ressourcen, Zuverlässigkeit, Unabhängigkeit
    - Einstiegsknoten werden aus Menge der Entry Guards gewählt
    - Solange Entry Guards nicht vom Angreifer übernommen, ist TOR-Route sicher

Paul Syverson et al.: Towards an Analysis of Onion Routing Security. Workshop on Design Issues in Anonymity and Unobservability, 2000

Route Selection-Angriff

- Auswahl von Tor-Routen entsprechend freier Übertragungskapazität
  - Tor-Knoten geben dafür selbst freie Kapazität an
- Route Selection Angriff
  - einschleusen von kompromittierten Tor-Knoten mit sehr vielen freien Ressourcen (oder Manipulation dieser Angaben)
  - diese Knoten werden von Tor bevorzugt genutzt → weniger Rechner können viele Daten abfangen



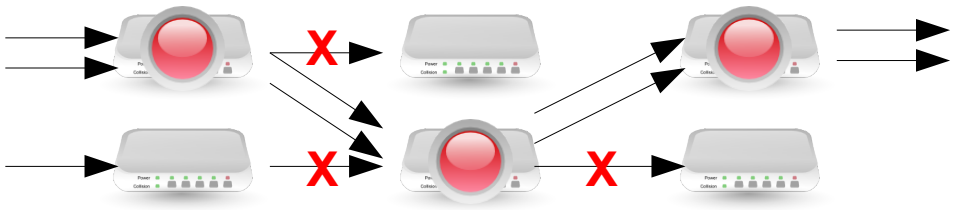
- Lösung: Messung der freien Kapazität „von außen“
  - Aber: Sicherheit fraglich; kompromittierte Rechner könnten sich gegenseitig validieren

Øverlier, L. et al.: Locating hidden servers. Proceedings of the IEEE Symposium on Security and Privacy, 2006

Selective Disruption-Angriff

- Eine Tor-Route ist sicher, solange nicht alle Knoten auf einem Pfad kompromittiert sind

- Selective Disruption-Angriff
  - Ein Teil aller Router im Tor-Netz ist kompromittiert
  - Ein kompromittierter Router stört absichtlich alle Pfade, die ehrliche Router enthalten
  - Tor baut neue Routen auf
    → Wahrscheinlichkeit, dass eine Route nur noch kompromittierte Knoten enthält, steigt



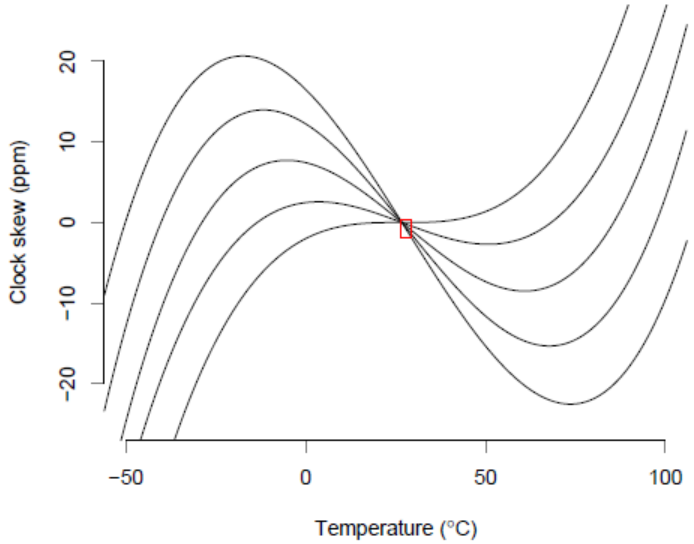
Kevin Bauer et al.:On the Optimal Path Length for Tor.
Hot Topics in Privacy Enhancing Technologies (HotPETS), 2010

Clock Skew-Angriff

- TCP-Timestamps werden vom Network Stack des Betriebssystems genutzt,
  - um Roundtrip-Zeiten zu schätzen
  - um Fehler in der Reihenfolge von Paketsequenzen zu entdecken

    → werden von Firewalls durchgelassen und von Routern mit weitergeleitet

- Clock Skew-Angriff
  - Abweichungen der Zeit hängen von Hardware und sogar Temperatur des Rechners ab
  - → Fingerabdruck des Rechners; statistische Angriffe auf die Tor-Route möglich

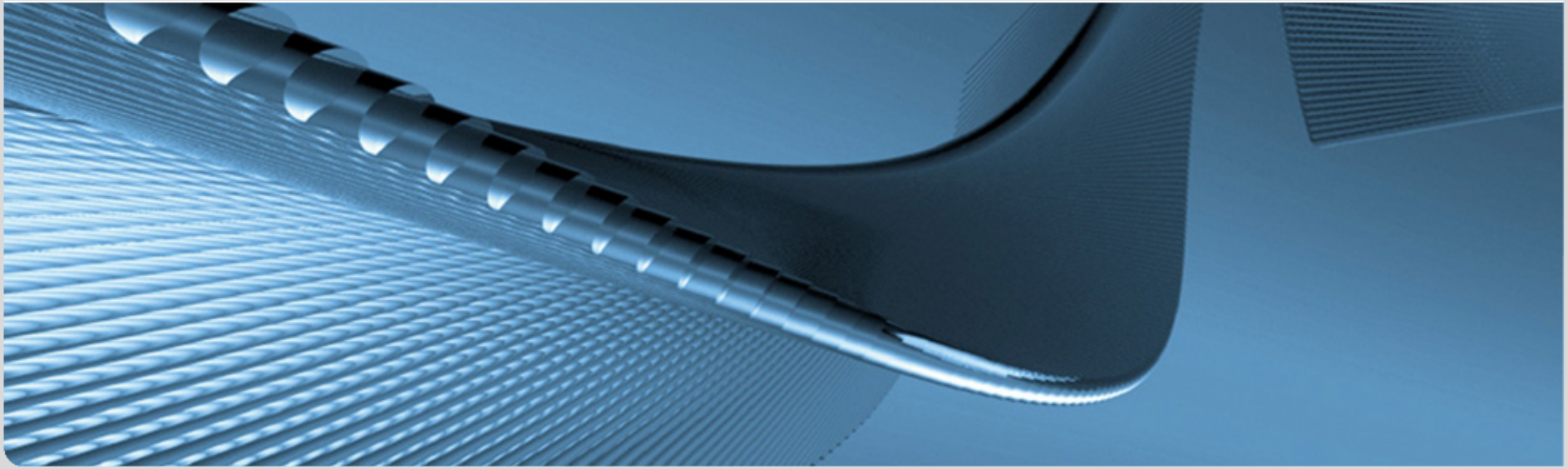


Stephen Murdoch et al.: Hot or not: Revealing hidden services by their clock skew, ACM Conference on Computer and Communications Security, 2006

Karlsruher Institut für Technologie

Freenet

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



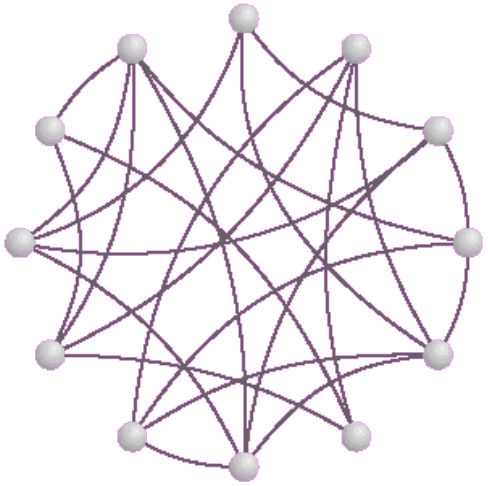
Freenet

- Ziel: **anonymer Datenaustausch, Zensurresistenz**
  - TOR Hidden Services: es existiert immernoch ein Server im Netz, der Daten speichert und einer Person zugeordnet ist
  - Freenet: *Abstreitbarkeit* für Informationsanbieter
    - Daten anonym und verteilt speichern, Peer-to-Peer Ansatz
    - keine Quelle, die man vom Netz nehmen könnte → Löschung und Zensur unmöglich

- entwickelt seit 2000, Open Source http://freenetproject.org
- *seit 2008 "Darknet" Feature, direkte Kommunikation nur mit vertrauenswürdigen Knoten*

Das Peer-to-Peer Modell

- viele *unabhängige* Knoten
- jeder Knoten kennt einige ausgewählte (strukturierte P2P-Systeme) oder zufällige (unstrukturierte Systeme) andere Knoten
  - lokales, unvollst. Wissen über Netzwerktopologie
- jeder Knoten leitet Nachrichten an einen anderen weiter, der in der Netzwerktopologie näher am Ziel ist
  - Small-World-Eigenschaft
    (vgl. Milgram-Experiment '69
    6 degrees of separation)



Freenet-Architektur

- Jeder Knoten speichert
  - Daten
  - Routing-Tabelle mit ausgewählten Knoten
    - IP-Adresse und gespeicherte Daten des Knotens
- Daten werden über **ortsunabhängige** Schlüssel referenziert
  - Daten dürfen zwischen den Knoten “umziehen”
  - häufig abgefragte Daten werden repliziert,
    unnötige gemäß Least-recently-used aus den Caches gelöscht
- Anfragen nach diesen Schlüsseln werden über Ketten von Freenet-Knoten weitergeleitet
  - Verschlüsselung zwischen den Knoten
  - Routing-Topologie lernt über die Zeit

Referenzierung über Schlüssel

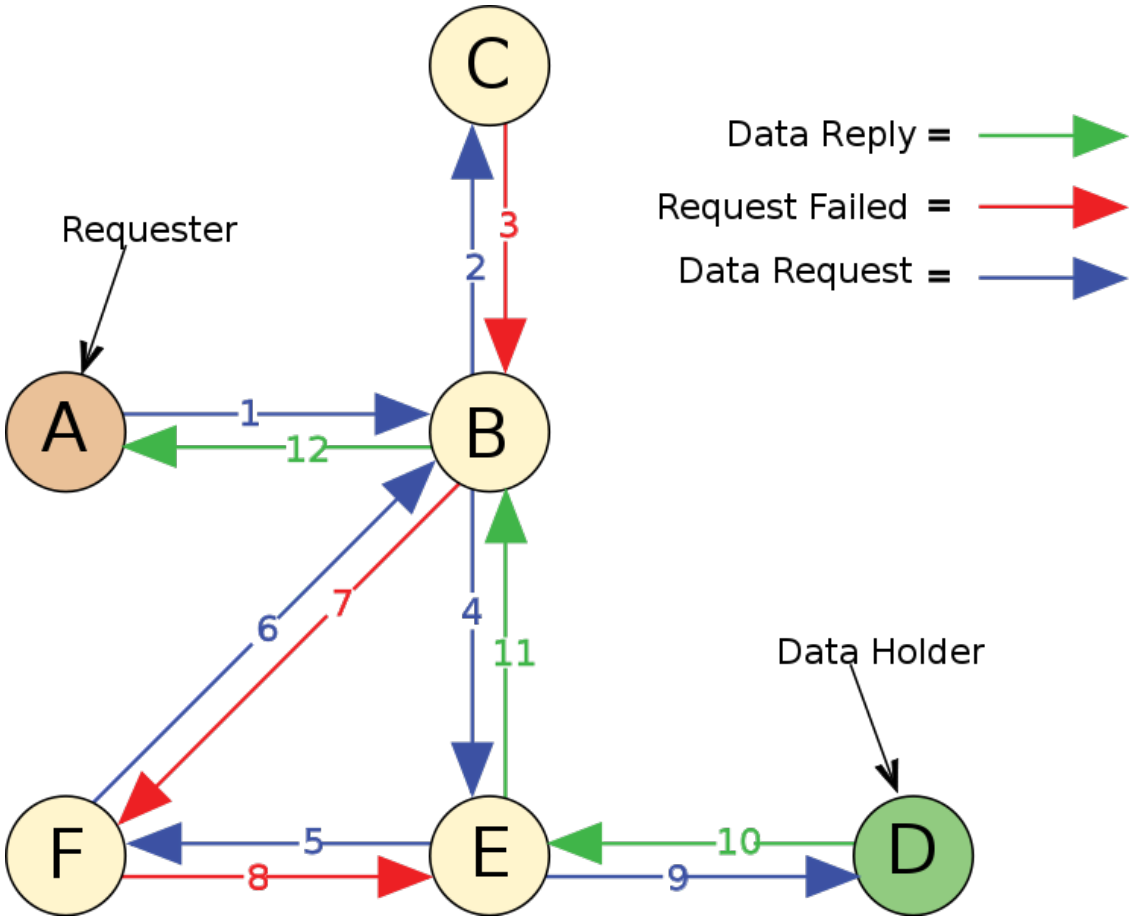
- Zwei relevante Schlüsseltypen
  (keyword-signed keys sind nicht mehr im Einsatz)
  - **Content-hash key**, krypt. Signatur der Daten
    - vergleichbar mit Inodes im Dateisystem
  - **Signed-subspace key**, Namespaces, in denen jeder lesen, aber nur der Ersteller schreiben kann
    - vergleichbar mit Datei- und Verzeichnisnamen
- Verbreitung der Schlüssel als Lesezeichen, in öffentlichen Directories oder über Freenet-Suchmaschinen

Daten abrufen (1/2)

1. Schlüssel beschaffen
2. Anfrage mit Schlüssel an einen Freenet-Knoten senden
3. Knoten erhält Anfragen
   - merken, von wem die Nachricht kam
   - zum Schlüssel passende Daten lokal gespeichert?
     - Ja: Antwort zurücksenden
     - Nein: Anfrage an den Knoten senden, dessen Schlüsselbereiche dem gesuchten Schlüssel am ähnlichsten sind (*nächste Folie*)
4. Knoten leitet Antwort zurück
   - Daten im eigenen Repository zwischenspeichern, anderen mitteilen, dass er nun diese Daten kennt
   - Routingtabelle updaten

Daten abrufen (2/2)

- Query Routing in Freenet: Hillclimbing + Backtracking



Anonymität in Freenet

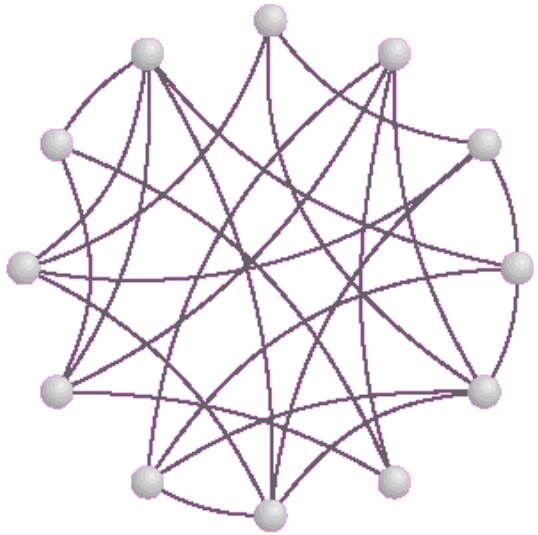
- Anfrager bleibt anonym
  - Nachricht sucht sich ihr Ziel anhand Schlüssel selbst,
  - Verbindungen nur jeweils zwischen zwei Knoten,
  - dem Schlüssel ist nicht anzusehen, was er bezeichnet
- Ersteller der Daten bleibt anonym
  - Daten werden durch ihre Schlüssel adressiert, und an beliebiger Stelle ins Freenet eingespeist
- Zensurresistenz
  - Daten können prinzipiell an beliebigen Stellen gespeichert werden
  - beliebte Daten häufig in den Caches der Knoten
  - Daten werden nur verschlüsselt abgespeichert, d.h. Freenet-Betreiber weiß nicht welche Daten er hostet

Das Darknet-Feature in Freenet

- Grundsätzliches Problem in allen anonymen Netzen:
  das Einschleusen von manipulierten Knoten
  - mit wenigen Rechnern/Ressourcen gezielt angreifen, wenn “die richtigen Stellen” in der Netztopologie bekannt/erraten
- Darknet
  - abgeschlossenes, privates Friend-to-Friend-Netzwerk
  - keine Kommunikation mit unbekannten Knoten,
    daher Einschleusen erschwert
  - Herausforderung: trotzdem ein verbundenes Netzwerk schaffen

Umsetzung von Darknet

- Routingtabellen werden nicht mehr selbstlernend von den Knoten angelegt, sondern von Hand
  - Betreiber kennen sich
- Implementierung eines strukturierten Overlays in Form eines Ringes, auf das Schlüssel im Interval [0,1] abgebildet werden
- Knoten können Plätze im Ring tauschen
  - es bilden sich Cluster von Knoten, die
    - effizient kommunizieren und
    - ähnliche Daten speichern

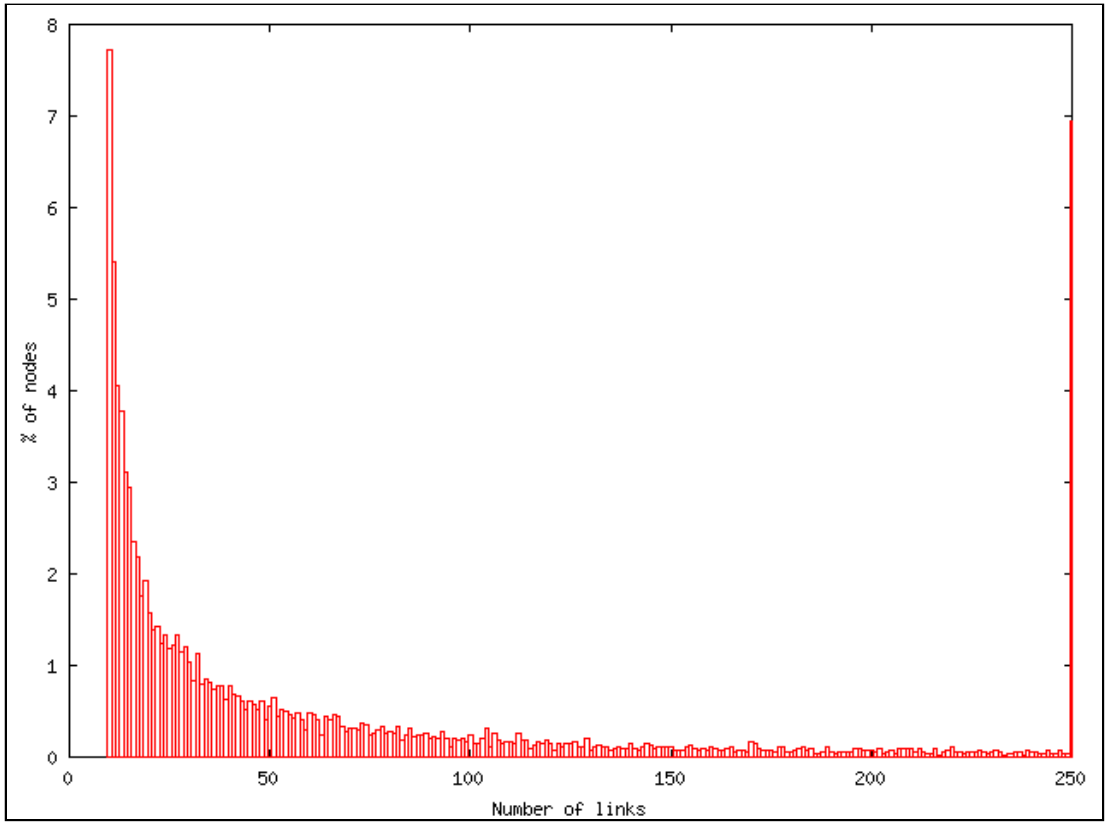


Grenzen des Ansatzes (1/2)

- Keine Garantien
  - Routing-Verfahren kann Daten nicht finden, obwohl sie im Netz gespeichert sind
  - selten gesuchte Daten können verschwinden (Least-Recently-Used-Ansatz)
  - der erste gefragte Knoten kann gleichzeitig Ersteller und Speicherort der Daten sein
    → erfährt unmittelbar, wer etwas wissen will
- Skalierbarkeit bzg. Zahl der Knoten, Pfadlänge
  - Small-World-Ansatz ist wenig effizient;
    strukturierte P2P-Systeme wären besser
    → Darknet-Erweiterung löst das Problem teilweise
- Darknet erfordert, andere Darknet-Nutzer zu kennen

Grenzen des Ansatzes (2/2)

- Skalenfreies Netz, einige wenige Knoten wichtiger als andere
  → Angriffspunkte!

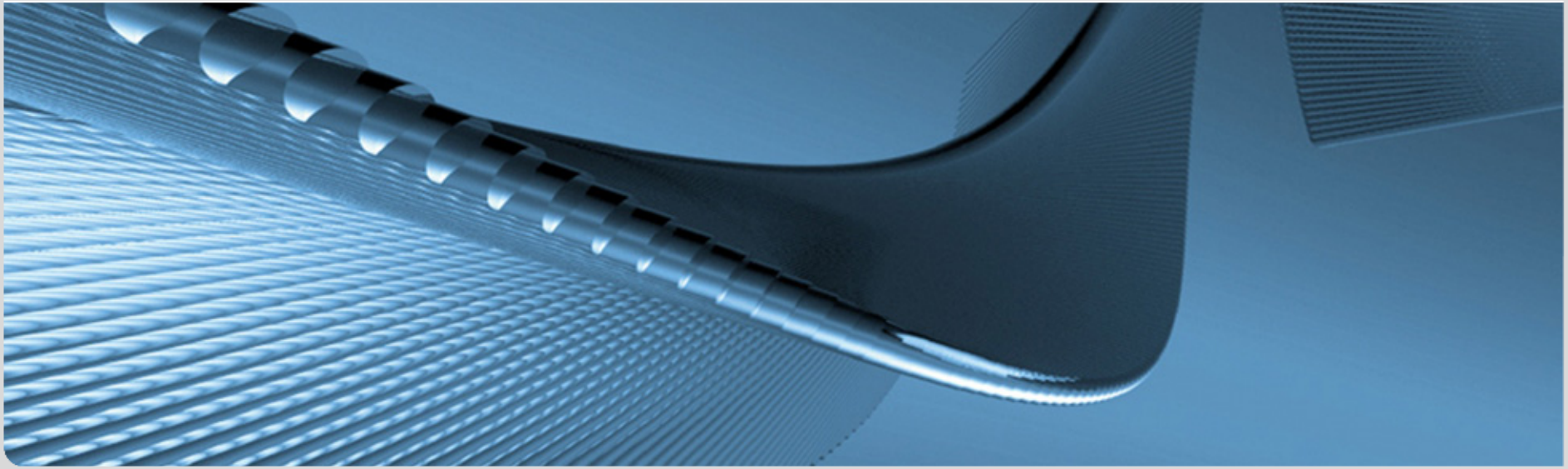


Quelle: Topics in Reliable Distributed Computing, Yoav Levy 2004

Karlsruher Institut für Technologie

Zusammenfassung

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Zusammenfassung

- Die IP-Adresse ist unvermeidlich, jedoch aus Datenschutzsicht problematisch
  - Quasi-Identifier
  - ggf. Zuordnung zur Person möglich
- Datenschutzansätze basierend auf dem Konzept von Mixen:
  - JAP
  - TOR
  - Freenet

Literatur

[1] Roger Dingledine et al.: *Tor: The Second-Generation Onion Router*, Proceedings of the SSYM 2004
http://freehaven.net/tor/tor-design.pdf

[2] Ian Clarke et al.; *Freenet: A Distributed Anonymous Information Storage and Retrieval System*, LNCS 2009
http://citeseer.ist.psu.edu/old/clarke00freenet.html