RPKI-RTR Client Library

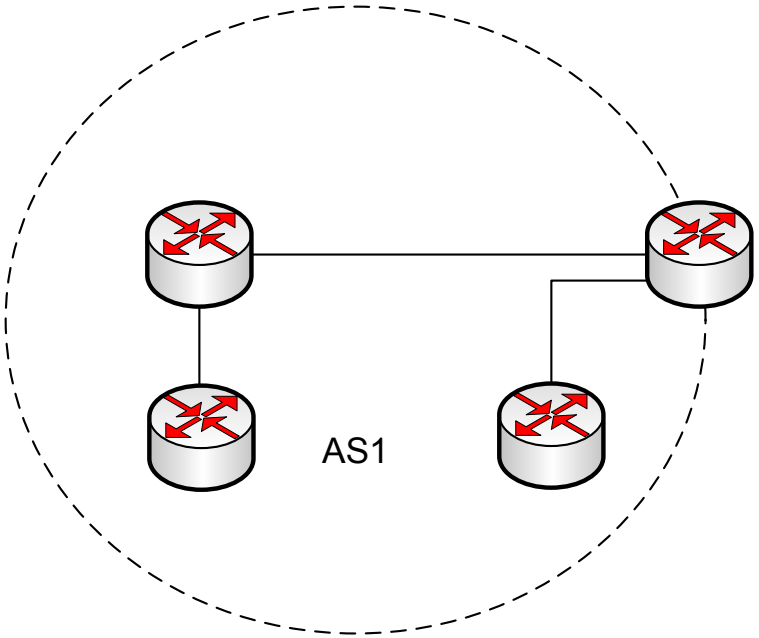
Fabian Holler

HAW Hamburg

1. Juni 2011

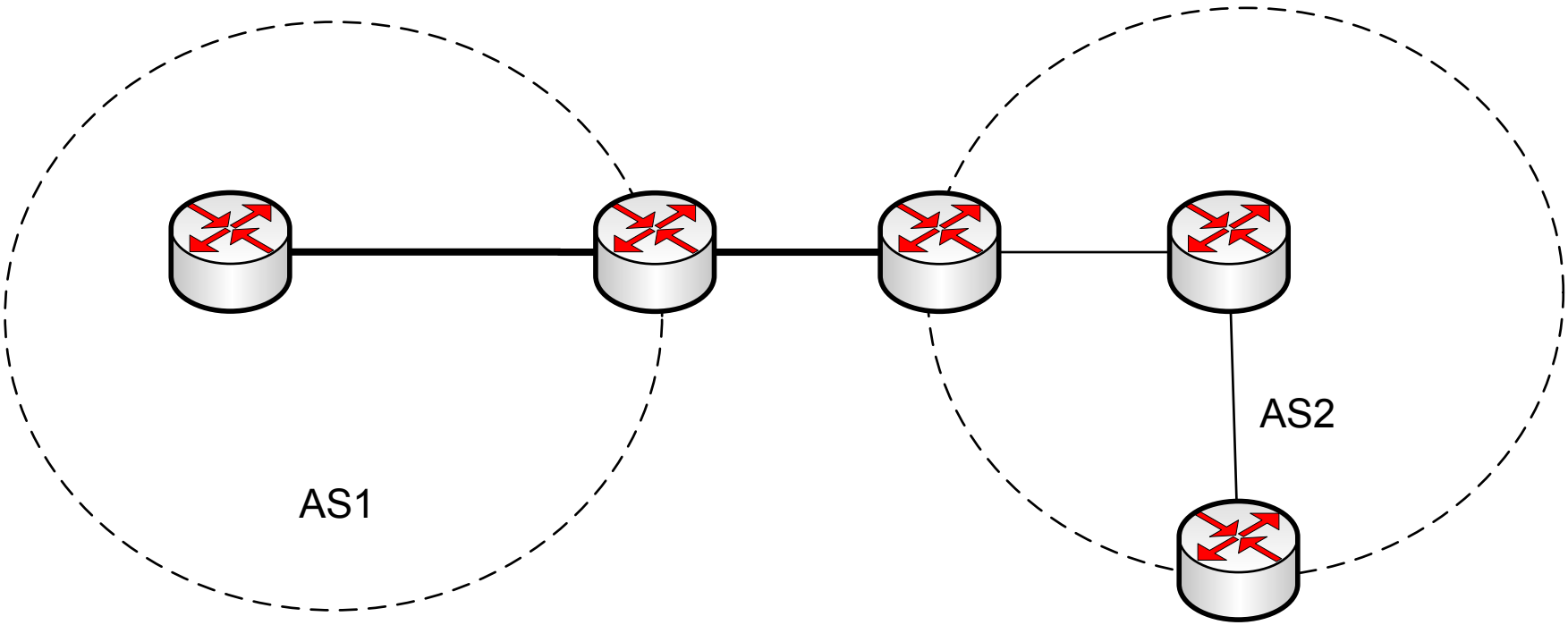
Agenda

Autonomes System



- Organisationseinheit
- Umfasst Menge von IP-Netze
- Identifikation durch autonomous system number (ASN)

Autonome Systeme



- Austausch von Routing-Informationen zwischen ASes
  - Interior-Gateway-Protokolle (IGP) innerhalb von ASes
  - Exterior-Gateway-Protokolle (EGP) zwischen ASes

Border Gateway Protocol

- De-facto Standard EG-Protokoll
- BGP1: 1989 spezifiziert
- BGP4: Seit 1994 im Einsatz [6]
- wenige Sicherheits-/Robustheitsmechanismen

Border Gateway Protocol

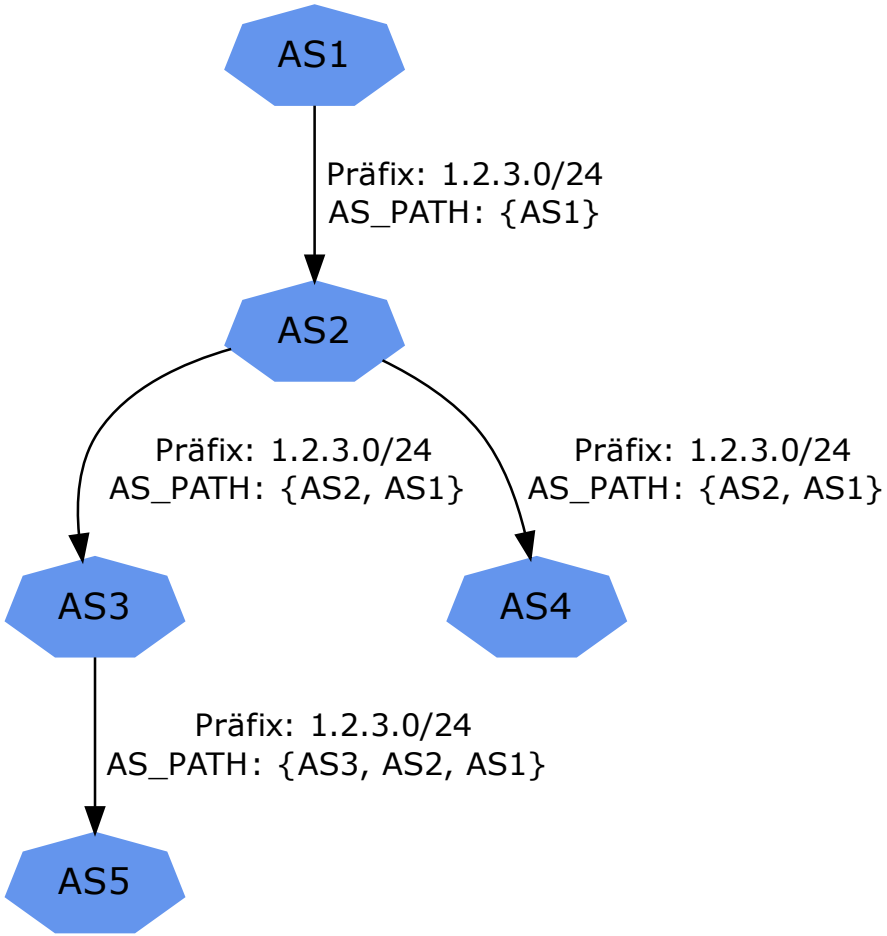
- Nachrichtenaustausch über TCP (Port 179)
- Verbindung mit statisch konfigurierten BGP-Neighbors
- Inkrementeller Datenaustausch
- Routing-Informationen werden über BGP-UPDATE Nachrichten übertragen

BGP-UPDATE

BGP-UPDATE
Withdrawn Routes Length
Withdrawn Routes: (Prefix, Length, . . .)
Total Path Attribute Length
Path Attributes
Network Layer Reachability Information (Prefix, Length, . . .)

- **Path Attributes:**
- **AS_PATH:** Pfad des Announcement
  - Origin_AS
  - Metrik
  - Schleifenerkennung
- **NEXT_HOP:** IP-Adresse des Gateway für die annoncierten Präfixe [5]

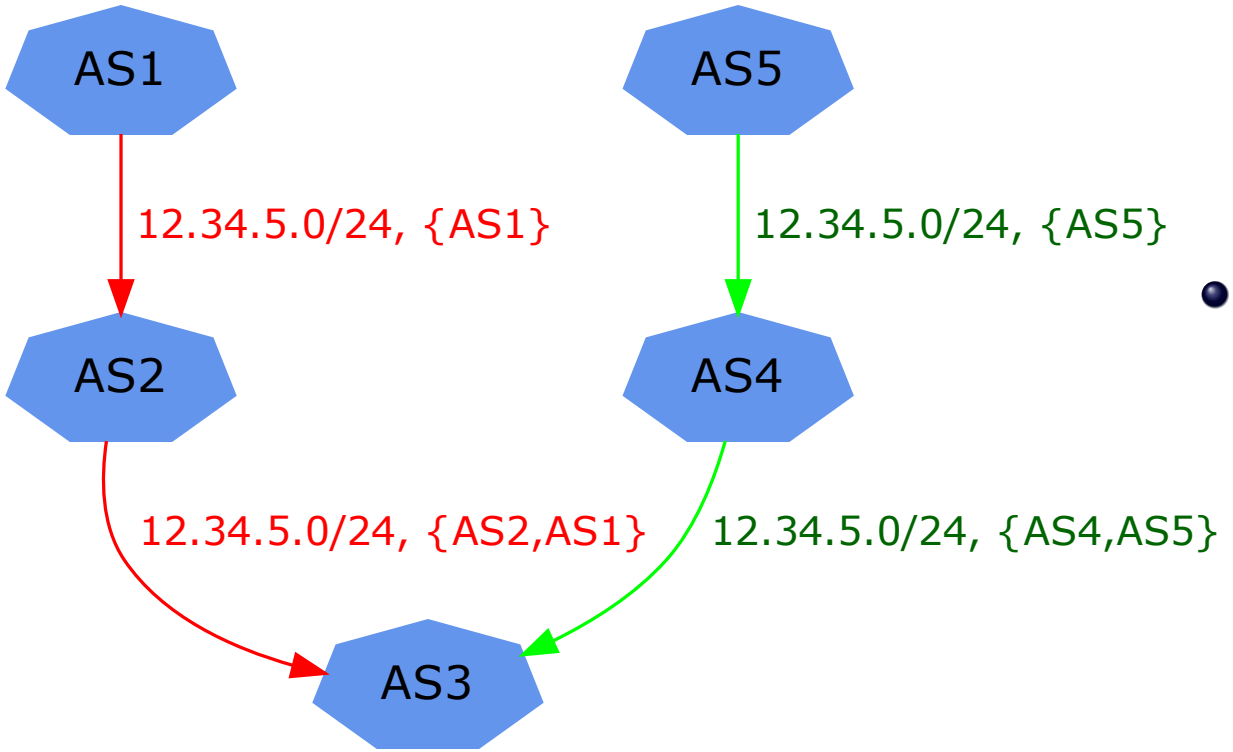
BGP Routen-Announcement



BGP Präfix-Hijacking

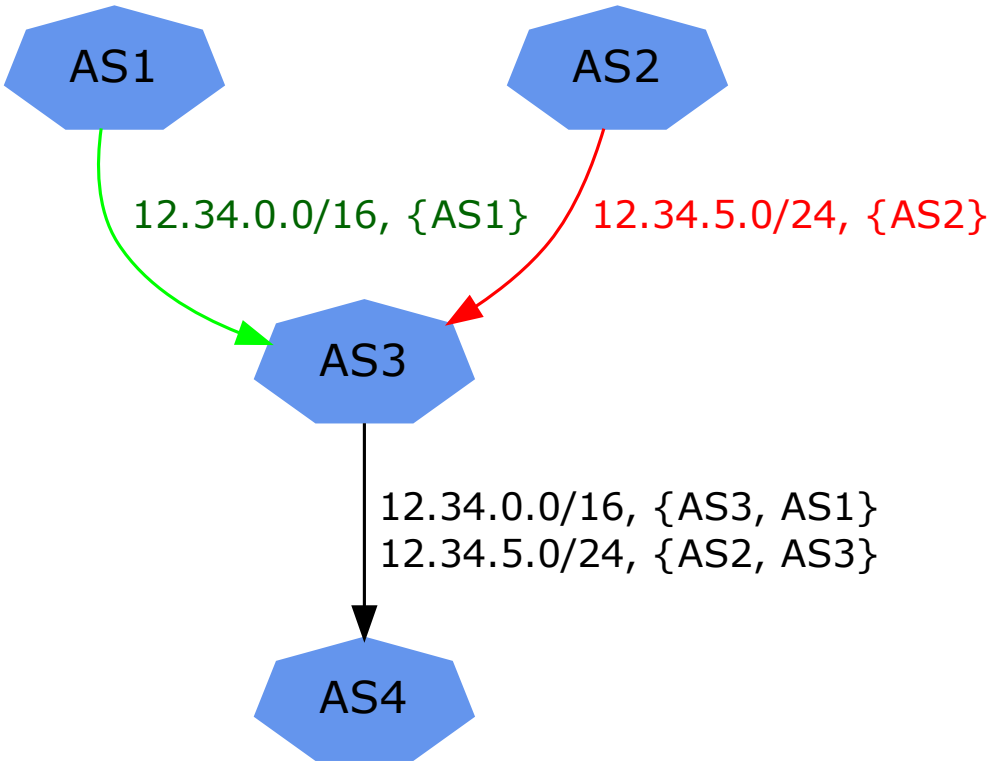
- BGP-Peer annonciert Route für Präfix welches ihm nicht gehört
- Keine Überprüfung der Validität der Route möglich
- BGP-Neighbors entscheiden anhand lokaler Regeln ob Route akzeptiert wird

BGP Präfix-Hijacking



- Kürzerer AS_PATH wird bevorzugt

BGP Präfix-Hijacking: Deaggregation



- Longest-Prefix-Match bevorzugt AS2 für 12.34.5.0/24
- Verbreitung im kompletten Internet wahrscheinlich

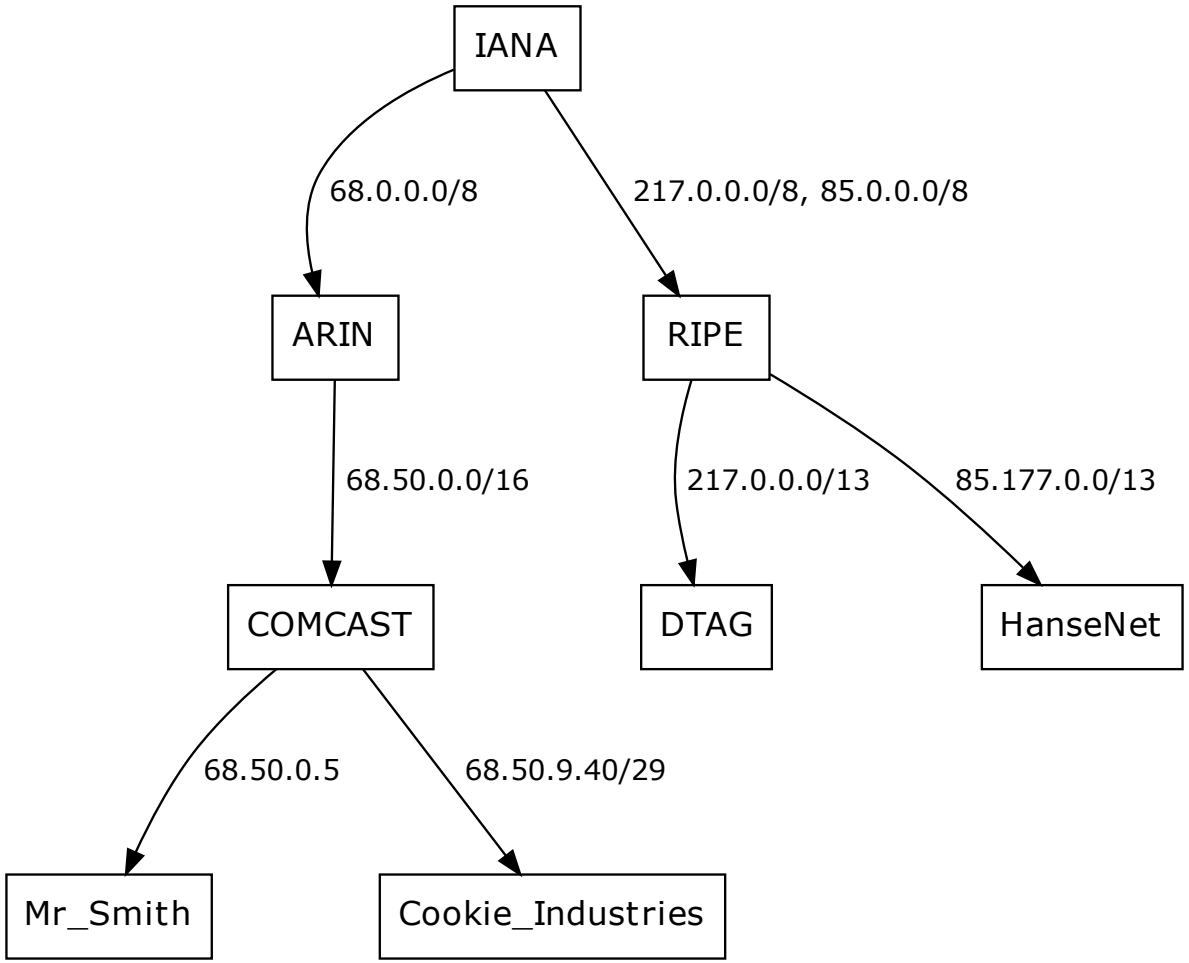
BGP Präfix-Hijacking Vorfälle

- 24.02.08 Pakistan Telecom hijackt YouTube.com Netzbereich [1]
  - Präfix Deaggregation
- 08.04.10 China Telecom annonciert Routen für 37.000 Präfixe statt 40 [2]

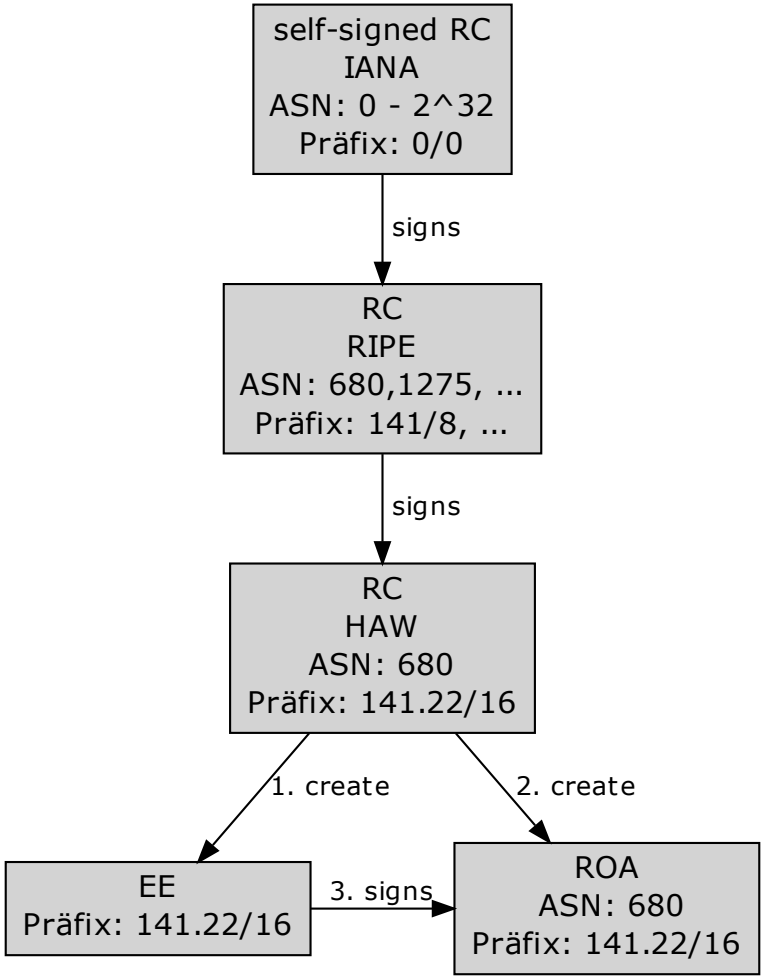
Origin Authentifizerung mit einer RPKI

- IETF Entwurf der Secure Inter-Domain Routing (sidr) Arbeitsgruppe
  - Idee basiert auf Secure BGP (S-BGP)
- Kryptographische Verifizierung des Herkunfts-AS von Routen-Annoncierungen
  - Ziel: Verhinderung von Präfix-Hijacking
- PKI bildet IP-/AS-Vergabestruktur ab

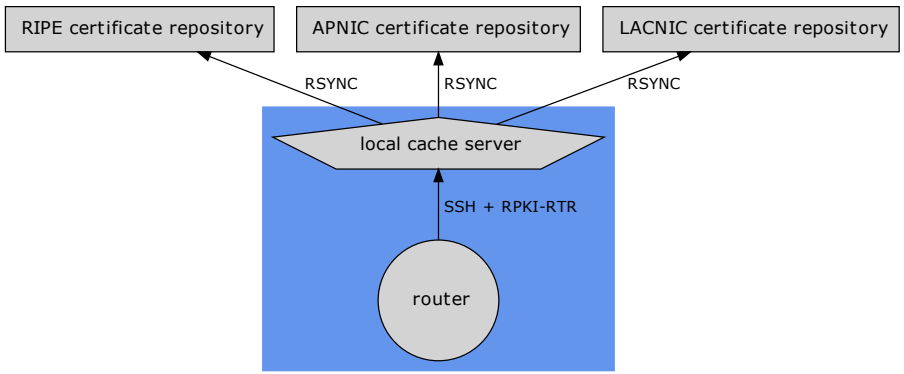
IP-Adressvergabe



RPKI Zertifikatskette



RPK-Infrastructure



- Zertifikate zugänglich über Repository Server der RIRs
- Validierung der Zertifikate auf lokalen Cache-Servern
- Cache-Servern stellen Liste mit validen Präfix/ASN Kombinationen bereit
- Kommunikation zwischen Router und Cache-Server mithilfe des RPKI-RTR-Protokolls

RPKI Einbindung in BGP-Router

- Router empfängt Liste mit validen Präfix/ASN Kombinationen über RPKI-RTR
- Bei Empfang einer neuen Route fragt BGP-Router Validierungsstatus ab
  - Validierungsstatus kann Preference Value der Route beeinflussen
  - Router kann Annoncierung mit ungültigem Validierungsstatus ignorieren

Fazit

- Minimale Zusatzlast auf Routern, durch Validierung auf lokalen Cache Servern
- Keine Änderung des BGP nötig
- Inkrementelle Installation möglich
- Verhindert Präfix-Hijacking
- Zahlreiche andere BGP-Angriffe nach wie vor durchführbar

RPKI-RTR

- Ziel: Entwicklung einer frei verfügbaren RPKI-RTR Client Library
- Cisco, Juniper arbeiten an Integration in Router
- Verbreitete freie BGP-Software besitzt noch keine RPKI-RTR Integration
  - Quagga
  - BIRD
  - OpenBGP
- Nutzung von SSH als Transportmedium
  - Authentizität des Cache-Servers
  - Sichere Datenübertragung
  - Kann sich in Zukunft ändern

RPKI-RTR

- Konfiguration:
- Liste von Server
  - Preference-Value
  - Hostname, Port, Username
  - Public SSH-Key vom Server
  - Private SSH-Key vom Client
- Lokale Clientdaten:
  - Seriennummer: Identifiziert Stand der Datensätze
  - Serial Nonce: Identifiziert Session
  - pfx_table: speichert valide AS/Präfix Kombinationen

RPKI-RTR

- Verbindungsaufbau
  - Client baut Verbindung mit Servern mit niedrigsten Preference Wert auf
  - Client sendet Reset-Query
  - Server sendet Cache-Response
    - ★ Cache-Nonce (ID der Client-session)
  - Server sendet aktuellen Datenbestand (IPv4/6 Präfix PDUs)
    - ★ ASN
    - ★ IP-Präfix
    - ★ IP-Präfix Länge
    - ★ IP-Präfix max. Länge
  - Server sendet End-of-data
    - ★ enthält aktuelle Seriennummer des aktuellen Datenbestandes

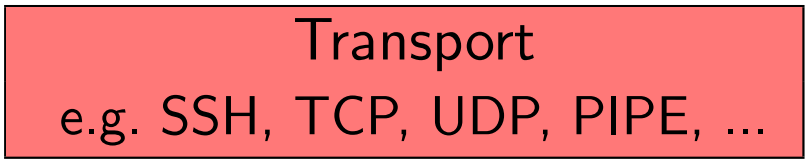
RPKI-RTR

- Datenabgleich
  - Client sendet nach Ablauf eines Timers eine Serial-Query PDU
  - Server sendet bei Datenänderung Serial-Notify

RPKI-RTR

- Übertragene Validierungsdatensätze von verbundenen Caches werden in pfx_table [4] gespeichert
  - ASN
  - Präfix
  - Prefix length
  - Prefix max. length
- Lookup Ergebnisse:
  - BGP_PFXV_STATE_VALID: Eintrag mit selbem Prefix, Len, maxLen, ASN
  - BGP_PFXV_STATE_INVALID: Präfix existiert in pfx_table, aber Len, maxLen oder ASN stimmen nicht überein
  - BGP_PFXV_STATE_NOT_FOUND: Präfix existiert nicht in pfx_table

RPKI-RTR Client Library Architektur



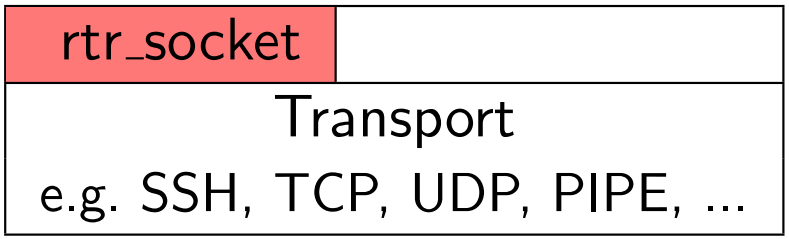
- Transport
  - Einheitl. Schnittstelle für versch. Transportprotokolle
  - Erweiterbar um Unterstützung für neue Transportprotokolle

Transport Socket

```
typedef struct tr_socket{
    const char* tech_name;
    void* socket;
    open_tr_fp open_fp;
    close_tr_fp close_fp;
    free_tr_fp free_fp;
    send_tr_fp send_fp;
    recv_tr_fp recv_fp;
} tr_socket;
```

- tr_socket.socket Datentyp wird durch spezielle Implementierung festgelegt

RPKI-RTR Client Library Architektur

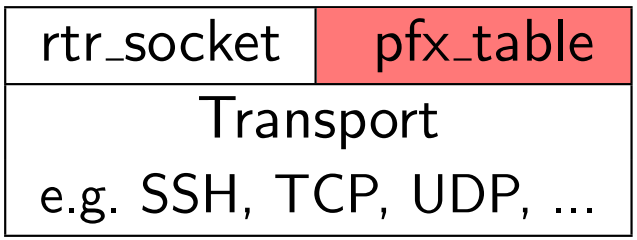


- rtr_socket
  - Implementiert RTR-Protokoll
  - Nutzt Transport Socket zur Datenübertragung

rtr_socket

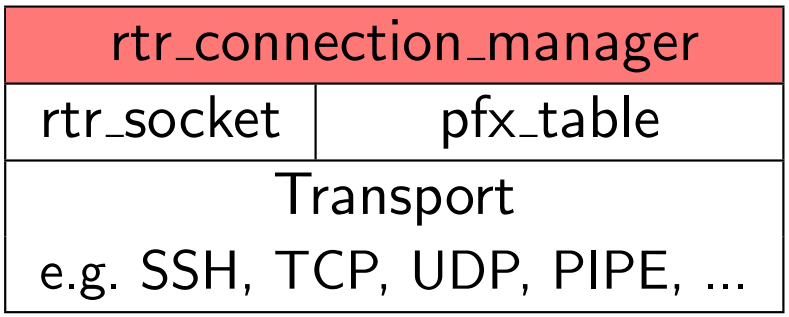
- Als StateMachine implementiert
  - ```
    typedef enum rtr_socket_state{
        RTR_CLOSED,
        RTR_ESTABLISHED,
        RTR_RESET,
        RTR_SYNC,
        RTR_ERROR_NO_DATA_AVAIL,
        RTR_ERROR_INCR_UPDATE_AVAIL,
        RTR_ERROR_FATAL
    } rtr_socket_state;
    ```
  - Bei Statusänderung wird konfigurierbare Callback-Funktion aufgerufen
    ```

RPKI-RTR Client Library Architektur



- pfx_table
  - Verwaltet Validierungsdatensätze von Cache-Servern
  - Stellt Operationen zur Präfix-Validierung bereit

RPKI-RTR Client Library Architektur



- rtr_connection_manager
  - Koordiniert mehrere rtr_sockets
  - Verbindungsaufbau in Reihenfolge der Preference Werte
  - Behandlung von Fehlerzustände, mithilfe Verbindungsstatus-Callbacks der rtr_sockets

Beispiel

```
#include <stdlib.h>
#include "transport/ssh/ssh_transport.h"
#include "transport/tcp/tcp_transport.h"
#include "rtr/rtr.h"

int main(){
    tr_socket* ssh_socket;
    tr_ssh_config config = {
        "141.22.26.248",
        "22",
        "fho",
        "/tmp/hostkey",
        "/tmp/id_ra",
        "/tmp/id_rsa.pub"
    };
    tr_ssh_init(&config, &ssh_socket);

    tr_socket* tcp_socket;
    tr_tcp_config tcp_config = {
        "141.22.26.248",
        "1234"
    };
    tr_tcp_init(&tcp_config, &tcp_socket);
```

Beispiel

```
    pfx_table pfx_t;

    //int rtr_init(tr_socket, polling_period, cache_timeout, rtr_socket, pfx_table,
         update_cb, rtr_update_cb_len, rtr_connection_state_cb,
         rtr_connection_state_cb_len);
    rtr_socket rtr_ssh;
    rtr_init(ssh_socket, 10, 360, &rtr_ssh, &pfx_t, NULL, 0, NULL, 0);
    rtr_socket rtr_tcp;
    rtr_init(tcp_socket, 10, 360, &rtr_tcp, &pfx_t, NULL, 0, NULL, 0);

    rtr_connection_pool p1{
        2,          //preference
        &rtr_tcp, //rtr_socket
        NULL        //next
    };
    rtr_connection_pool p0{
        3,          //preference
        &rtr_ssh, //rtr_socket
        &p1       //next
    };
    rtr_connection_pool_socket pool_sock;
    rtr_connection_manager_start(&p0, &pool_sock);

    pfxv_state state =  rtr_connection_manager_validate(&pool_sock, 3512, "12.34.0.0",
         16);
}
```

Vielen Dank für ihre Aufmerksamkeit

 "Youtube hijacking: A ripe ncc ris case study," 2008. [Online]. Available: http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

 "Chinese isp hijacks the internet," 2010. [Online]. Available: http://bgpmon.net/blog/?p=282

 K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *Proc. of the IEEE*, vol. 98, no. 1, pp. 100–122, January 2010.

 P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein, "BGP Prefix Origin Validation," IETF, Internet-Draft – work in progress 01, February 2011.

 Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task

Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

I. van Beijnum, *BGP - building reliable networks with the border gateway protocol*. O'Reilly, 2002. [Online]. Available: http://www.oreilly.de/catalog/bgp/index.html