net2o: Transportschicht — Implementiert

Zähme das Netz

Bernd Paysan

Forth–Tagung 2012, Biezenmortel

Übersicht

Motivation

Datenflusssteuerung

Zuverlässigkeit

Kryptographie

Was ist am Internet kaputt

- TCP–Flow Control: Stichwort „Buffer Bloat“
- TCP als „rundum–Sorglos“–Protokoll leider nicht mal halbwegs echtzeitfähig, also doch nicht „rundum sorglos“
- UDP ist nur ein „einfacher Zugang“ zu IP, und ansonsten „do it yourself“
- Die SSL–PKI, mit den „ehrlichen Achmeds“ als Certification Authorities
- Verschlüsselung insgesamt „zu komplex, zu schwierig,“ weshalb das meiste überhaupt nicht verschlüsselt wird

Was ist am Internet kaputt

- TCP–Flow Control: Stichwort „Buffer Bloat“
- TCP als „rundum–Sorglos“–Protokoll leider nicht mal halbwegs echtzeitfähig, also doch nicht „rundum sorglos“
- UDP ist nur ein „einfacher Zugang“ zu IP, und ansonsten „do it yourself“
- Die SSL–PKI, mit den „ehrlichen Achmeds“ als Certification Authorities
- Verschlüsselung insgesamt „zu komplex, zu schwierig,“ weshalb das meiste überhaupt nicht verschlüsselt wird

Was ist am Internet kaputt

- TCP–Flow Control: Stichwort „Buffer Bloat“
- TCP als „rundum–Sorglos“–Protokoll leider nicht mal halbwegs echtzeitfähig, also doch nicht „rundum sorglos“
- UDP ist nur ein „einfacher Zugang“ zu IP, und ansonsten „do it yourself“
- Die SSL–PKI, mit den „ehrlichen Achmeds“ als Certification Authorities
- Verschlüsselung insgesamt „zu komplex, zu schwierig,“ weshalb das meiste überhaupt nicht verschlüsselt wird

Was ist am Internet kaputt

- TCP–Flow Control: Stichwort „Buffer Bloat“
- TCP als „rundum–Sorglos“–Protokoll leider nicht mal halbwegs echtzeitfähig, also doch nicht „rundum sorglos“
- UDP ist nur ein „einfacher Zugang“ zu IP, und ansonsten „do it yourself“
- Die SSL–PKI, mit den „ehrlichen Achmeds“ als Certification Authorities
- Verschlüsselung insgesamt „zu komplex, zu schwierig,“ weshalb das meiste überhaupt nicht verschlüsselt wird

Was ist am Internet kaputt

- TCP–Flow Control: Stichwort „Buffer Bloat“
- TCP als „rundum–Sorglos“–Protokoll leider nicht mal halbwegs echtzeitfähig, also doch nicht „rundum sorglos“
- UDP ist nur ein „einfacher Zugang“ zu IP, und ansonsten „do it yourself“
- Die SSL–PKI, mit den „ehrlichen Achmeds“ als Certification Authorities
- Verschlüsselung insgesamt „zu komplex, zu schwierig,“ weshalb das meiste überhaupt nicht verschlüsselt wird

Änderungen gegenüber dem Entwurf

- Paketgröße jetzt $64 * 2^n$, $n \in \{0, \ldots, 15\}$, also bis zu 2MB in Zweierpotenzen
- Kein embedded–Protokoll implementiert, nur die 64–Bit–Variante
- Verschlüsselung ist immer aktiv, nicht optional
- Kein „Salt“ am Anfang des verschlüsselten Pakets, dafür ein Hash (128 Bits) am Ende

Änderungen gegenüber dem Entwurf

- Paketgröße jetzt $64 * 2^n, \quad n \in \{0, \ldots, 15\}$, also bis zu 2MB in Zweierpotenzen
- Kein embedded–Protokoll implementiert, nur die 64–Bit–Variante
- Verschlüsselung ist immer aktiv, nicht optional

  Kein „Salt“ am Anfang des verschlüsselten Pakets, dafür ein Hash (128 Bits) am Ende

Änderungen gegenüber dem Entwurf

- Paketgröße jetzt $64 * 2^n, \quad n \in \{0, \ldots, 15\}$, also bis zu 2MB in Zweierpotenzen
- Kein embedded–Protokoll implementiert, nur die 64–Bit–Variante
- Verschlüsselung ist immer aktiv, nicht optional

  Kein „Salt“ am Anfang des verschlüsselten Pakets, dafür ein Hash (128 Bits) am Ende

Änderungen gegenüber dem Entwurf

- Paketgröße jetzt $64 * 2^n, \quad n \in \{0, \ldots, 15\}$, also bis zu 2MB in Zweierpotenzen
- Kein embedded–Protokoll implementiert, nur die 64–Bit–Variante
- Verschlüsselung ist immer aktiv, nicht optional
- Kein „Salt“ am Anfang des verschlüsselten Pakets, dafür ein Hash (128 Bits) am Ende

Status: TCP Flow Control

- TCP füllt erst mal den Puffer voll, bis ein Paket verloren geht, statt vorher schon einzugreifen. Beschreibung der Lage: „Buffer bloat“

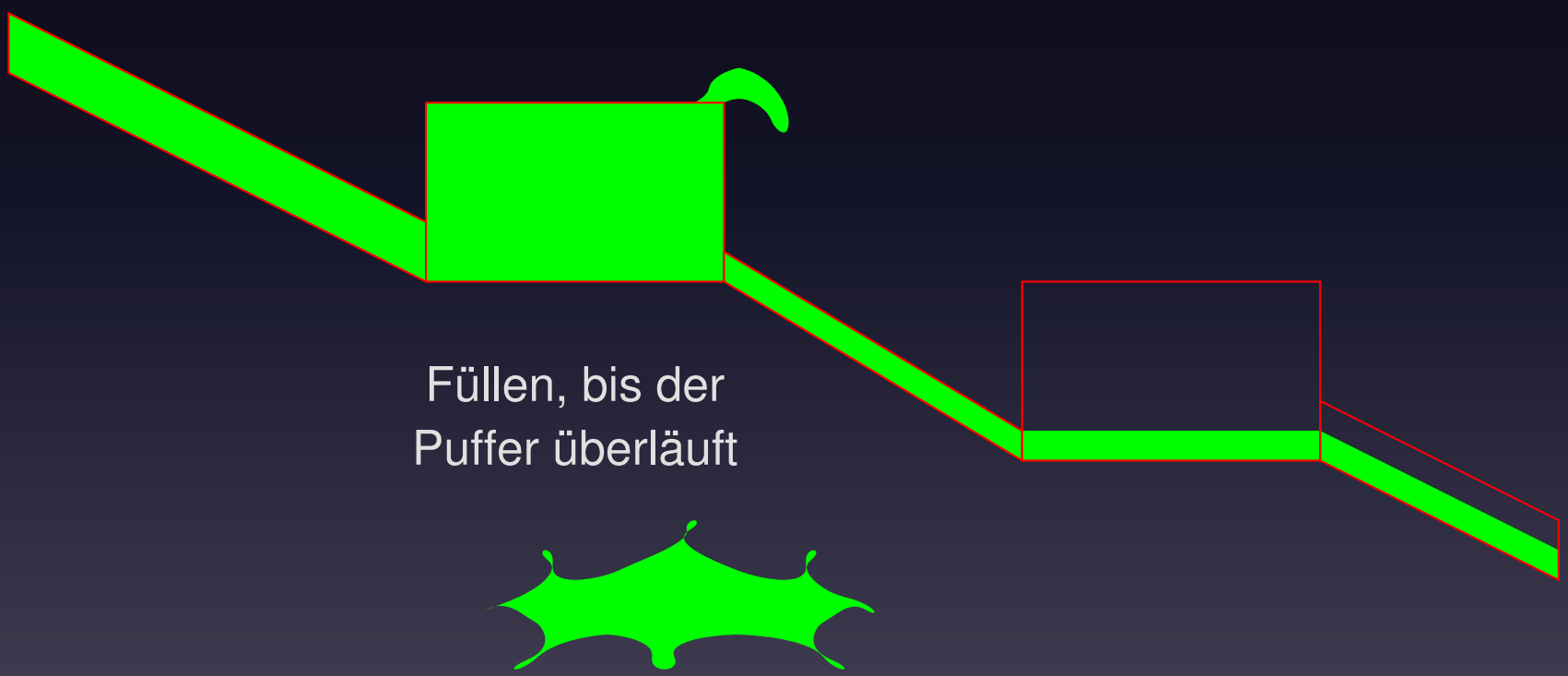

Abbildung: Buffer Bloat

Alternativen?

- LEDBAT versucht, einen konstanten Delay hinzuzufügen: Funktioniert nur sehr beschränkt — keine faire Konkurrenz
- CurveCP hat auch einen Ansatz, der aber nicht dokumentiert ist. . .
- Also muss etwas neues her

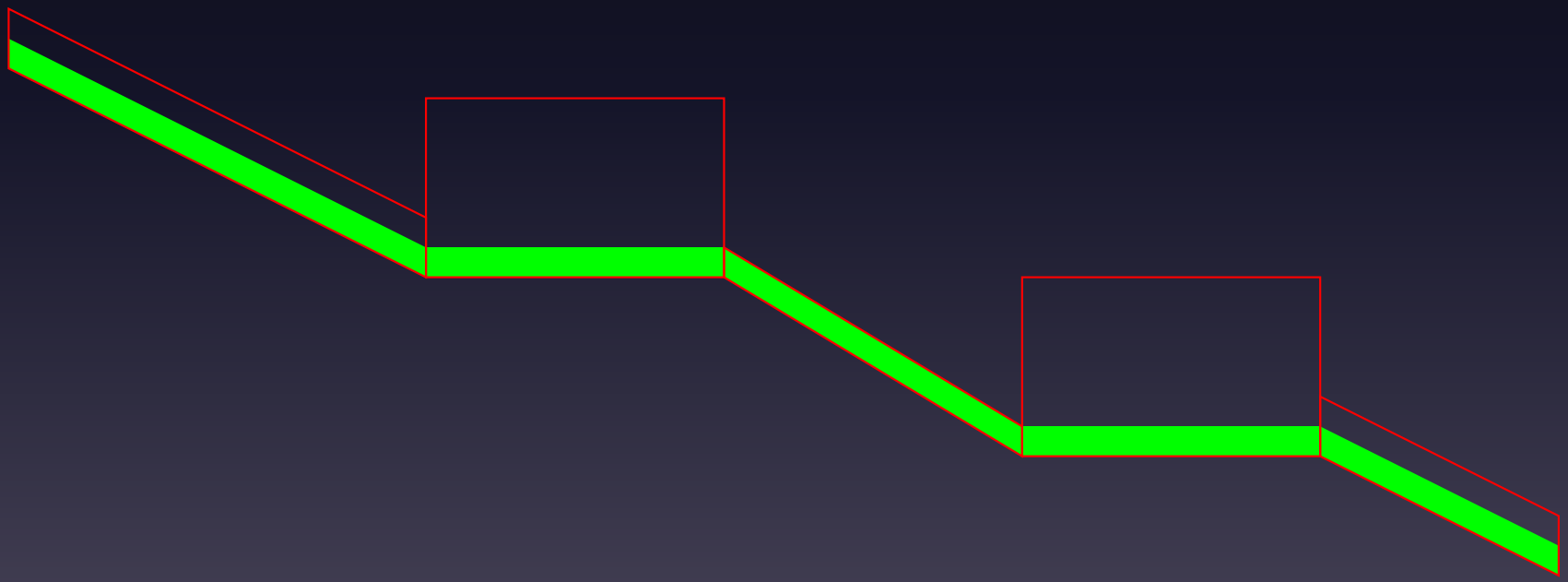

Abbildung: So sollte das aussehen

Alternativen?

- LEDBAT versucht, einen konstanten Delay hinzuzufügen: Funktioniert nur sehr beschränkt — keine faire Konkurrenz
- CurveCP hat auch einen Ansatz, der aber nicht dokumentiert ist. . .
- Also muss etwas neues her

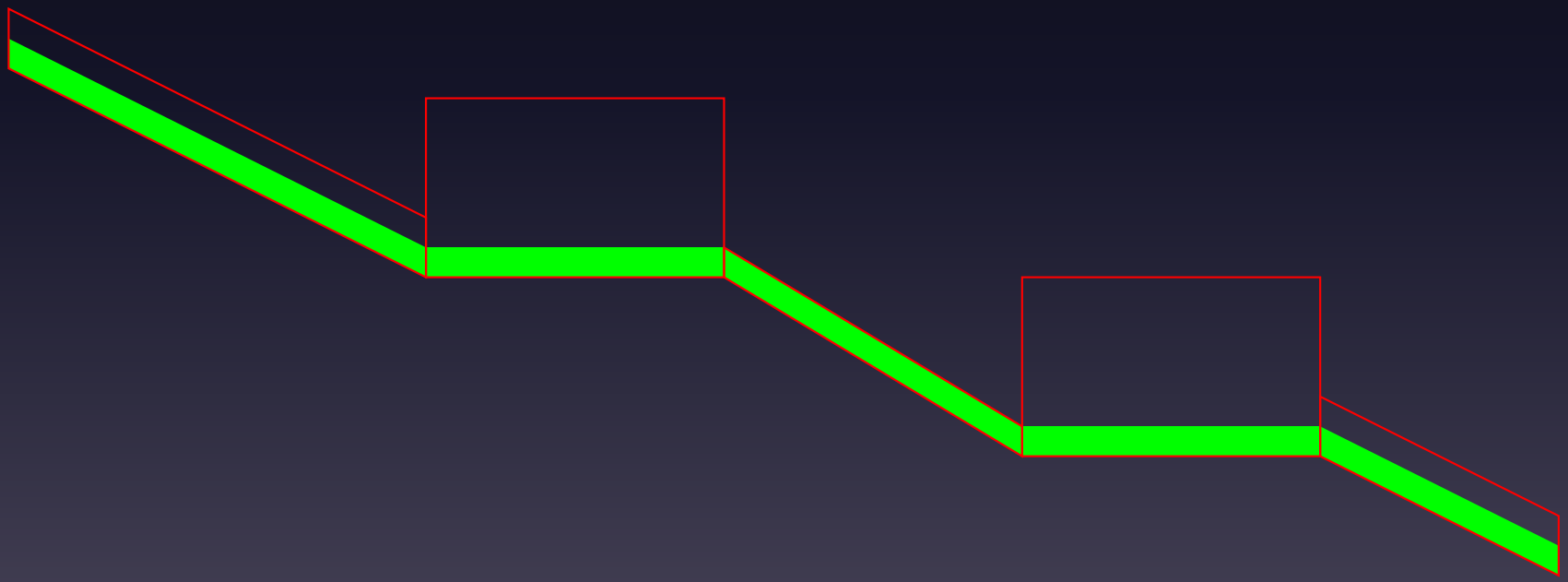

Abbildung: So sollte das aussehen

Alternativen?

- LEDBAT versucht, einen konstanten Delay hinzuzufügen: Funktioniert nur sehr beschränkt — keine faire Konkurrenz
- CurveCP hat auch einen Ansatz, der aber nicht dokumentiert ist. . .
- Also muss etwas neues her

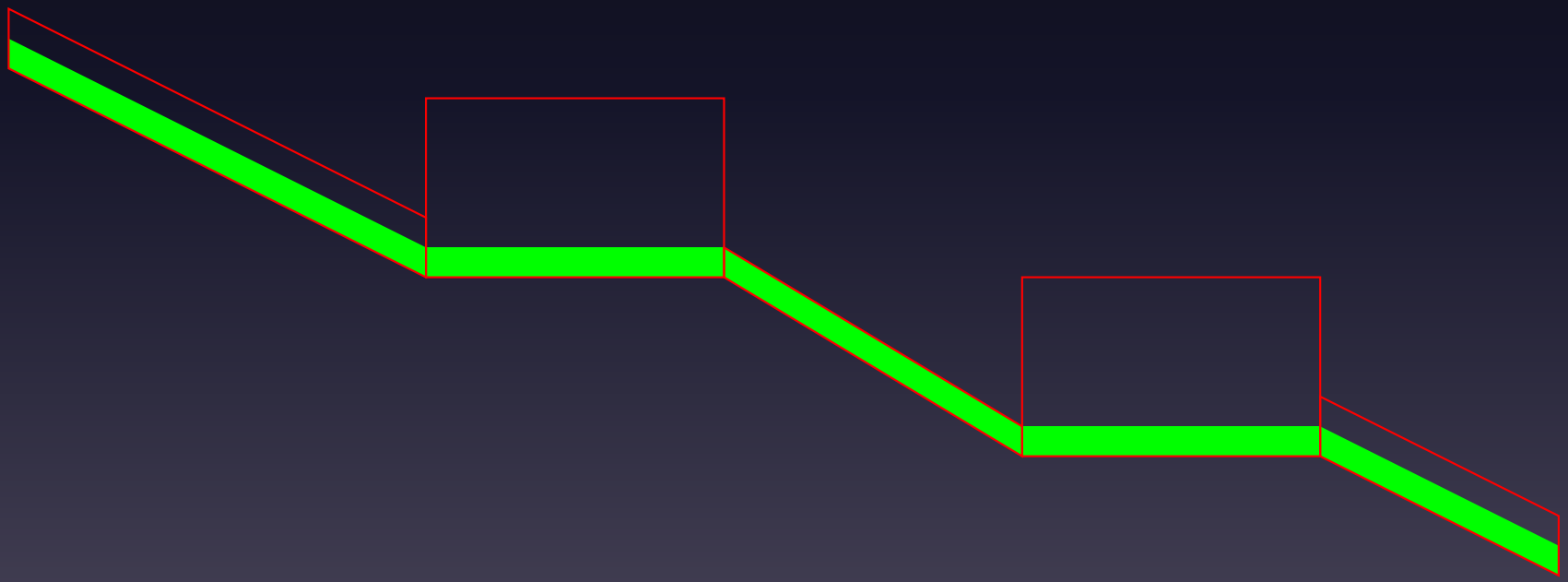

Abbildung: So sollte das aussehen

„Buffer Bloat“

- Retransmits kosten Netzwerkresourcen, sollen vermieden werden
- Wie groß muss der Buffer sein, unter der Annahme, dass die Bandbreite optimal genutzt wird, der Flaschenhals am anderen Ende der Verbindung ist und ein zweiter Datenstrom dazugeschaltet wird?

  Antwort: Ca. die Hälfte des Round–Trip–Delays, denn die werden unausweichlich gefüllt
- Buffer sind gut, man muss sie ja nicht bis zum Rand füllen!

„Buffer Bloat“

- Retransmits kosten Netzwerkresourcen, sollen vermieden werden
- Wie groß muss der Buffer sein, unter der Annahme, dass die Bandbreite optimal genutzt wird, der Flaschenhals am anderen Ende der Verbindung ist und ein zweiter Datenstrom dazugeschaltet wird?

  Antwort: Ca. die Hälfte des Round–Trip–Delays, denn die werden unausweichlich gefüllt
- Buffer sind gut, man muss sie ja nicht bis zum Rand füllen!

„Buffer Bloat“

- Retransmits kosten Netzwerkresourcen, sollen vermieden werden
- Wie groß muss der Buffer sein, unter der Annahme, dass die Bandbreite optimal genutzt wird, der Flaschenhals am anderen Ende der Verbindung ist und ein zweiter Datenstrom dazugeschaltet wird?
- Antwort: Ca. die Hälfte des Round–Trip–Delays, denn die werden unausweichlich gefüllt
- Buffer sind gut, man muss sie ja nicht bis zum Rand füllen!

„Buffer Bloat“

- Retransmits kosten Netzwerkresourcen, sollen vermieden werden
- Wie groß muss der Buffer sein, unter der Annahme, dass die Bandbreite optimal genutzt wird, der Flaschenhals am anderen Ende der Verbindung ist und ein zweiter Datenstrom dazugeschaltet wird?
- Antwort: Ca. die Hälfte des Round–Trip–Delays, denn die werden unausweichlich gefüllt
- Buffer sind gut, man muss sie ja nicht bis zum Rand füllen!

net2o Flow Control

Abbildung: Ausmessen des Flaschenhalses mit einem Burst

Client misst, Server setzt Rate

Client erfasst die *Zeit* des ersten und letzten Pakets im Burst, und zählt die Anzahl der Pakete nach dem ersten. Daraus ergibt sich die gewünschte Rate:

```
: calc-rate ( - )
  delta-ticks @ tick-init 1+ acks @ */
  lit, set-rate ;
```

Server berücksichtigt noch den Slack (also den Füllstand des Buffers), um weiter abzubremsen:

```
: set-rate ( rate - )
  lastdiff @ min-slack @ -
  0 max slack# 4 << min slack# @ / lastdiff @
  ns/burst ! ;
```

Client misst, Server setzt Rate

Client erfasst die *Zeit* des ersten und letzten Pakets im Burst, und zählt die Anzahl der Pakete nach dem ersten. Daraus ergibt sich die gewünschte Rate:

```
: calc-rate ( - )
  delta-ticks @ tick-init 1+ acks @ */
  lit, set-rate ;
```

Server berücksichtigt noch den Slack (also den Füllstand des Buffers), um weiter abzubremsen:

```
: set-rate ( rate - )
  lastdiff @ min-slack @ -
  0 max slack# 4 * min  slack# / lshift
  ns/burst ! ;
```

Test über VDSL

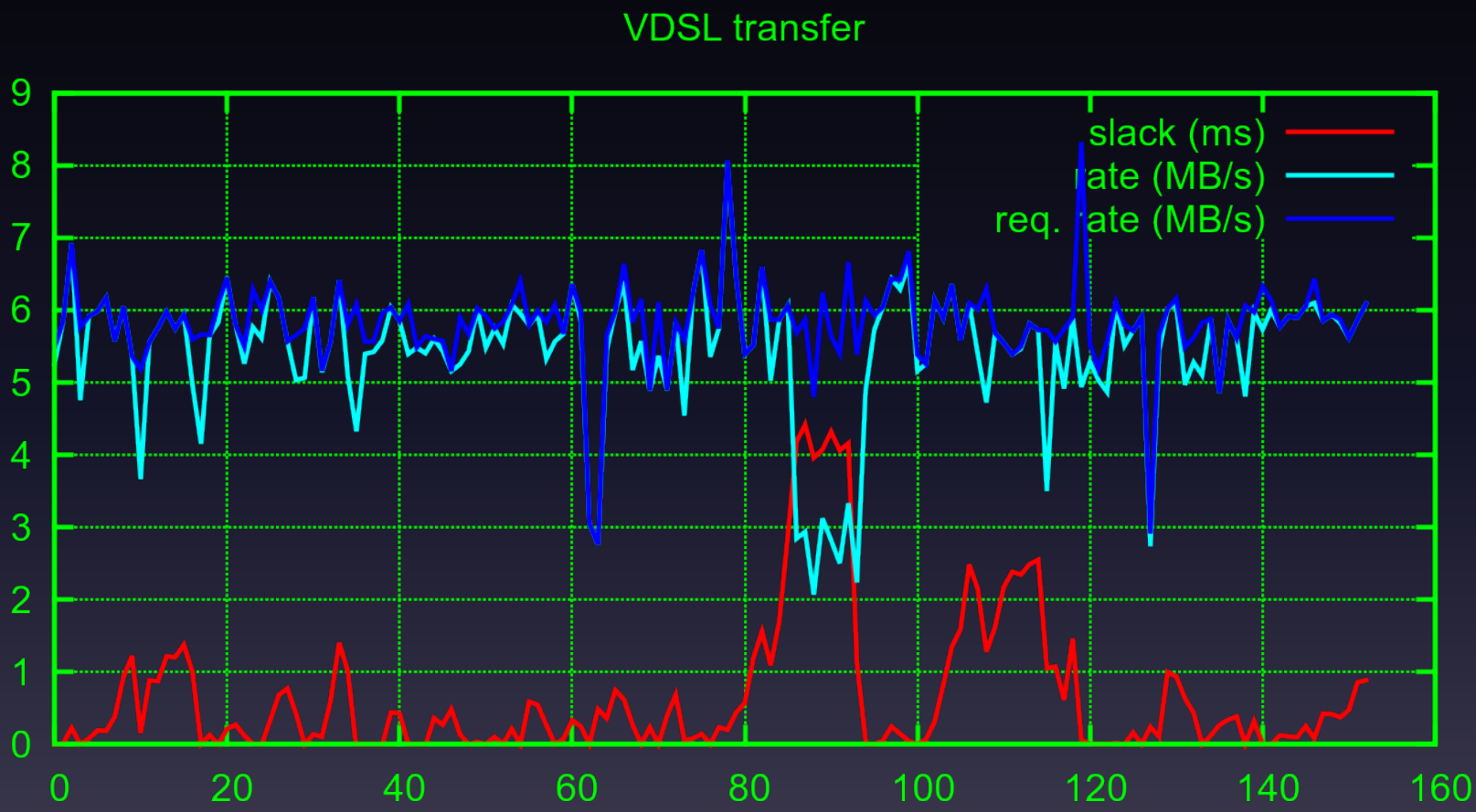

Abbildung: Beispiel–Transport über VDSL

Unzuverlässiges Luftkabel

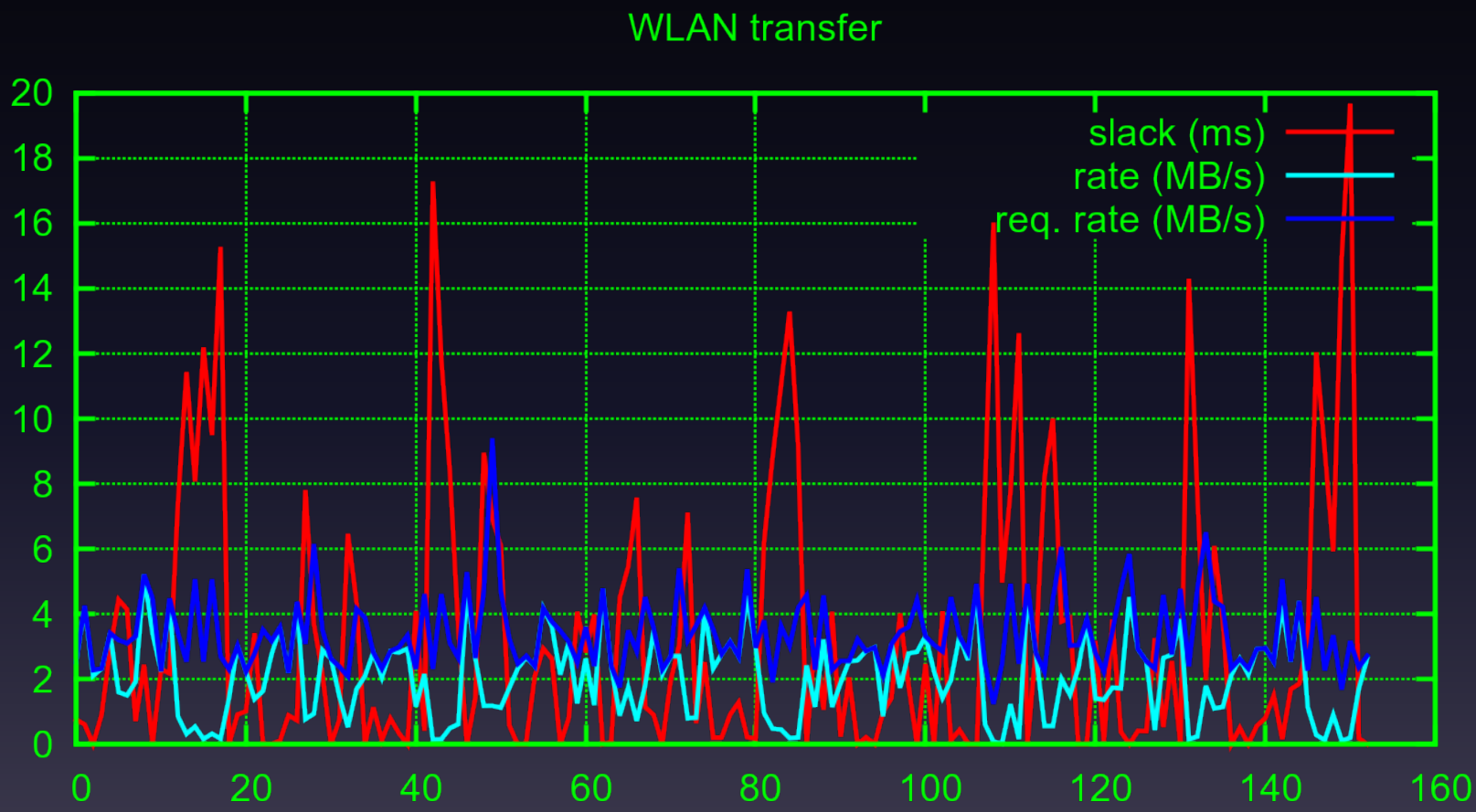

Abbildung: Beispiel–Transport über VDSL+WLAN

Mehrere parallele Verbindungen

- Der Burst erkennt nur die Geschwindigkeit für eine Verbindung — das liegt daran, dass die existierenden Buffers als FIFO für den gesamten Verkehr implementiert sind. Besser wäre: Parallele FIFOs, in der jede Verbindung getrennt behandelt wird, und dann round robin zwischen diesen FIFOs.
- Exponentielle Verlangsamung durch Beobachtung des aufgebauten Slacks hilft
- Lösung zwar fair, aber noch nicht stabil — es werden reihum Bandbreiten vergeben
- Manchmal ist der Slack so groß, dass es zu Paketverlusten kommt (Ausgangsproblem)
- Noch etwas mehr Finetuning nötig!

Mehrere parallele Verbindungen

- Der Burst erkennt nur die Geschwindigkeit für eine Verbindung — das liegt daran, dass die existierenden Buffers als FIFO für den gesamten Verkehr implementiert sind. Besser wäre: Parallele FIFOs, in der jede Verbindung getrennt behandelt wird, und dann round robin zwischen diesen FIFOs.
- Exponentielle Verlangsamung durch Beobachtung des aufgebauten Slacks hilft
- Lösung zwar fair, aber noch nicht stabil — es werden reihum Bandbreiten vergeben
- Manchmal ist der Slack so groß, dass es zu Paketverlusten kommt (Ausgangsproblem)
- Noch etwas mehr Finetuning nötig!

Mehrere parallele Verbindungen

- Der Burst erkennt nur die Geschwindigkeit für eine Verbindung — das liegt daran, dass die existierenden Buffers als FIFO für den gesamten Verkehr implementiert sind. Besser wäre: Parallele FIFOs, in der jede Verbindung getrennt behandelt wird, und dann round robin zwischen diesen FIFOs.
- Exponentielle Verlangsamung durch Beobachtung des aufgebauten Slacks hilft
- Lösung zwar fair, aber noch nicht stabil — es werden reihum Bandbreiten vergeben
- Manchmal ist der Slack so groß, dass es zu Paketverlusten kommt (Anzeige stockt)
- Noch etwas mehr Finetuning nötig!

Mehrere parallele Verbindungen

- Der Burst erkennt nur die Geschwindigkeit für eine Verbindung — das liegt daran, dass die existierenden Buffers als FIFO für den gesamten Verkehr implementiert sind. Besser wäre: Parallele FIFOs, in der jede Verbindung getrennt behandelt wird, und dann round robin zwischen diesen FIFOs.
- Exponentielle Verlangsamung durch Beobachtung des aufgebauten Slacks hilft
- Lösung zwar fair, aber noch nicht stabil — es werden reihum Bandbreiten vergeben
- Manchmal ist der Slack so groß, dass es zu Paketverlusten kommt (unerwünscht)
- Noch etwas mehr Finetuning nötig!

Mehrere parallele Verbindungen

- Der Burst erkennt nur die Geschwindigkeit für eine Verbindung — das liegt daran, dass die existierenden Buffers als FIFO für den gesamten Verkehr implementiert sind. Besser wäre: Parallele FIFOs, in der jede Verbindung getrennt behandelt wird, und dann round robin zwischen diesen FIFOs.
- Exponentielle Verlangsamung durch Beobachtung des aufgebauten Slacks hilft
- Lösung zwar fair, aber noch nicht stabil — es werden reihum Bandbreiten vergeben
- Manchmal ist der Slack so groß, dass es zu Paketverlusten kommt (unerwünscht)
- Noch etwas mehr Finetuning nötig!

Zuverlässigkeit des Transports

- Paket–Reihenfolge spielt keine Rolle, da die Pakete gemäß ihrer Adresse im Puffer abgelegt werden
- Angeforderte Pakete werden in zwei Bitmaps „nicht vorhanden“ markiert
- Empfangene Pakete werden in jeweils einer der beiden Bitmaps als empfangen markiert
- Die andere Bitmap wird aufgefüllt, wenn sie nicht für die Retransmits zuständig ist
- Es wird einen Round–Trip–Delay gewartet, bis ein Retransmit angefordert wird.
- Retransmit-Anforderungen werden gebündelt

Zuverlässigkeit des Transports

- Paket–Reihenfolge spielt keine Rolle, da die Pakete gemäß ihrer Adresse im Puffer abgelegt werden
- Angeforderte Pakete werden in zwei Bitmaps „nicht vorhanden“ markiert
- Empfangene Pakete werden in jeweils einer der beiden Bitmaps als empfangen markiert
- Die andere Bitmap wird aufgefüllt, wenn sie nicht für die Retransmits zuständig ist
- Es wird einen Round–Trip–Delay gewartet, bis ein Retransmit angefordert wird.
- Retransmit-Anforderungen werden [illegible]

Zuverlässigkeit des Transports

- Paket–Reihenfolge spielt keine Rolle, da die Pakete gemäß ihrer Adresse im Puffer abgelegt werden
- Angeforderte Pakete werden in zwei Bitmaps „nicht vorhanden“ markiert
- Empfangene Pakete werden in jeweils einer der beiden Bitmaps als empfangen markiert
- Die andere Bitmap wird aufgefüllt, wenn sie nicht für die Retransmits zuständig ist
- Es wird einen Round–Trip–Delay gewartet, bis ein Retransmit angefordert wird.
- Retransmit-Anforderungen werden gebündelt

Zuverlässigkeit des Transports

- Paket–Reihenfolge spielt keine Rolle, da die Pakete gemäß ihrer Adresse im Puffer abgelegt werden
- Angeforderte Pakete werden in zwei Bitmaps „nicht vorhanden“ markiert
- Empfangene Pakete werden in jeweils einer der beiden Bitmaps als empfangen markiert
- Die andere Bitmap wird aufgefüllt, wenn sie nicht für die Retransmits zuständig ist
- Es wird einen Round–Trip–Delay gewartet, bis ein Retransmit angefordert wird.
- Retransmit-Anforderungen werden

Zuverlässigkeit des Transports

- Paket–Reihenfolge spielt keine Rolle, da die Pakete gemäß ihrer Adresse im Puffer abgelegt werden
- Angeforderte Pakete werden in zwei Bitmaps „nicht vorhanden“ markiert
- Empfangene Pakete werden in jeweils einer der beiden Bitmaps als empfangen markiert
- Die andere Bitmap wird aufgefüllt, wenn sie nicht für die Retransmits zuständig ist
- Es wird einen Round–Trip–Delay gewartet, bis ein Retransmit angefordert wird.
- Retransmits werden bevorzugt behandelt.

Zuverlässigkeit des Transports

- Paket–Reihenfolge spielt keine Rolle, da die Pakete gemäß ihrer Adresse im Puffer abgelegt werden
- Angeforderte Pakete werden in zwei Bitmaps „nicht vorhanden“ markiert
- Empfangene Pakete werden in jeweils einer der beiden Bitmaps als empfangen markiert
- Die andere Bitmap wird aufgefüllt, wenn sie nicht für die Retransmits zuständig ist
- Es wird einen Round–Trip–Delay gewartet, bis ein Retransmit angefordert wird.
- Retransmits werden bevorzugt behandelt

Zuverlässiges Ausführen von Kommandos

Dieser Bereich ist noch eine Baustelle

- Das Kommando ist noch nicht ausgeführt worden $\longrightarrow$ ausführen, zugehörige Antwort merken
- Das Kommando ist schon ausgeführt worden $\longrightarrow$ zugehörige Antwort nochmal verschicken
- Nicht alle Kommandos werden beantwortet $\longrightarrow$ Was tun mit diesen?
- Acknowledges noch nicht fälschungssicher: Bedarf einer Kenntnis der empfangenen Pakete (Checksumme über die Checksummen plus Bitmap, auf welche Pakete sich das Acknowledge bezieht)

Zuverlässiges Ausführen von Kommandos

Dieser Bereich ist noch eine Baustelle

- Das Kommando ist noch nicht ausgeführt worden $\longrightarrow$ ausführen, zugehörige Antwort merken
- Das Kommando ist schon ausgeführt worden $\longrightarrow$ zugehörige Antwort nochmal verschicken
- Nicht alle Kommandos werden beantwortet $\longrightarrow$ Was tun mit diesen?
- Acknowledges noch nicht fälschungssicher: Bedarf einer Kenntnis der empfangenen Pakete (Checksumme über die Checksummen plus Bitmap, auf welche Pakete sich dies Acknowledge bezieht)

Zuverlässiges Ausführen von Kommandos

Dieser Bereich ist noch eine Baustelle

- Das Kommando ist noch nicht ausgeführt worden $\longrightarrow$ ausführen, zugehörige Antwort merken
- Das Kommando ist schon ausgeführt worden $\longrightarrow$ zugehörige Antwort nochmal verschicken
- Nicht alle Kommandos werden beantwortet $\longrightarrow$ Was tun mit diesen?
- Acknowledges noch nicht fälschungssicher: Bedarf einer Kenntnis der empfangenen Pakete (Checksumme über die Checksummen plus Bitmap, auf welche Pakete sich dies Acknowledge bezieht)

Zuverlässiges Ausführen von Kommandos

Dieser Bereich ist noch eine Baustelle

- Das Kommando ist noch nicht ausgeführt worden $\longrightarrow$ ausführen, zugehörige Antwort merken
- Das Kommando ist schon ausgeführt worden $\longrightarrow$ zugehörige Antwort nochmal verschicken
- Nicht alle Kommandos werden beantwortet $\longrightarrow$ Was tun mit diesen?
- Acknowledges noch nicht fälschungssicher: Bedarf einer Kenntnis der empfangenen Pakete (Checksumme über die Checksummen plus Bitmap, auf welche Pakete sich das Acknowledge bezieht)

Zuverlässiges Ausführen von Kommandos

Dieser Bereich ist noch eine Baustelle

- Das Kommando ist noch nicht ausgeführt worden $\longrightarrow$ ausführen, zugehörige Antwort merken
- Das Kommando ist schon ausgeführt worden $\longrightarrow$ zugehörige Antwort nochmal verschicken
- Nicht alle Kommandos werden beantwortet $\longrightarrow$ Was tun mit diesen?
- Acknowledges noch nicht fälschungssicher: Bedarf einer Kenntnis der empfangenen Pakete (Checksumme über die Checksummen plus Bitmap, auf welche Pakete sich das Acknowledge bezieht)

Kryptographie

Die ersten drei Ziele sollen bei der Kommunikation erfüllt werden, das vierte nicht:

Vertraulichkeit Dritte sollen nicht in der Lage sein, die Nachrichten abzuhören

Integrität Die Daten müssen nachweislich vollständig und unverändert sein.

Authentizität Der Absender der Nachrichten soll eindeutig identifizierbar sein

Verbindlichkeit ist nicht für die Kommunikation selbst wichtig — gegenüber dritten nachweisbare Urheberschaft ist für eine Ja-Wege-Kommunikation nicht nötig

Kryptographie

Die ersten drei Ziele sollen bei der Kommunikation erfüllt werden, das vierte nicht:

Vertraulichkeit Dritte sollen nicht in der Lage sein, die Nachrichten abzuhören

Integrität Die Daten müssen nachweislich vollständig und unverändert sein.

Authentizität Der Absender der Nachrichten soll eindeutig identifizierbar sein

Verbindlichkeit ist nicht für die Kommunikation selbst wichtig — gegenüber dritten nachweisbare Urheberschaft ist für eine Ja-Wette Kommunikation nicht nötig

Kryptographie

Die ersten drei Ziele sollen bei der Kommunikation erfüllt werden, das vierte nicht:

Vertraulichkeit Dritte sollen nicht in der Lage sein, die Nachrichten abzuhören

Integrität Die Daten müssen nachweislich vollständig und unverändert sein.

Authentizität Der Absender der Nachrichten soll eindeutig identifizierbar sein

Verbindlichkeit ist nicht für die Kommunikation selbst wichtig — gegenüber dritten nachweisbare Urheberschaft ist für eine Ja-Wage-Kommunikation nicht nötig

Kryptographie

Die ersten drei Ziele sollen bei der Kommunikation erfüllt werden, das vierte nicht:

Vertraulichkeit Dritte sollen nicht in der Lage sein, die Nachrichten abzuhören

Integrität Die Daten müssen nachweislich vollständig und unverändert sein.

Authentizität Der Absender der Nachrichten soll eindeutig identifizierbar sein

Verbindlichkeit ist nicht für die Kommunikation selbst wichtig — gegenüber dritten nachweisbare Urheberschaft ist für eine Ja-Wege-Kommunikation nicht nötig

Kryptographie

Die ersten drei Ziele sollen bei der Kommunikation erfüllt werden, das vierte nicht:

Vertraulichkeit Dritte sollen nicht in der Lage sein, die Nachrichten abzuhören

Integrität Die Daten müssen nachweislich vollständig und unverändert sein.

Authentizität Der Absender der Nachrichten soll eindeutig identifizierbar sein

Verbindlichkeit ist nicht für die Kommunikation selbst wichtig — gegenüber dritten nachweisbare Urheberschaft ist für eine 2–Wege–Kommunikation nicht nötig

Eingesetzte Technik: Curve25519

- Elliptic Curve Cryptography basiert nicht auf Faktorisierung, sondern auf natürlichen Logarithmen in elliptischen Kurven
- Die Sicherheit von Curve25519 entspricht 128 Bits eines symmetrischen Schlüssels — im Moment reicht das noch dicke.
- Curve25519 ist effizient implementiert
- Es ist für eine 1:1–Verbindung optimiert
- Jeder Teilnehmer „multipliziert“ seinen geheimen Schlüssel mit dem öffentlichen des Partners, die beiden Produkte

Eingesetzte Technik: Curve25519

- Elliptic Curve Cryptography basiert nicht auf Faktorisierung, sondern auf natürlichen Logarithmen in elliptischen Kurven
- Die Sicherheit von Curve25519 entspricht 128 Bits eines symmetrischen Schlüssels — im Moment reicht das noch dicke.
- Curve25519 ist effizient implementiert
- Es ist für eine 1:1–Verbindung optimiert
- Jeder Teilnehmer „multipliziert“ seinen geheimen Schlüssel mit dem öffentlichen des Partners, die beiden Produkte

Eingesetzte Technik: Curve25519

- Elliptic Curve Cryptography basiert nicht auf Faktorisierung, sondern auf natürlichen Logarithmen in elliptischen Kurven
- Die Sicherheit von Curve25519 entspricht 128 Bits eines symmetrischen Schlüssels — im Moment reicht das noch dicke.
- Curve25519 ist effizient implementiert
- Es ist für eine 1:1–Verbindung optimiert
- Jeder Teilnehmer „multipliziert“ seinen geheimen Schlüssel mit dem öffentlichen des Partners, die beiden Produkte

Eingesetzte Technik: Curve25519

- Elliptic Curve Cryptography basiert nicht auf Faktorisierung, sondern auf natürlichen Logarithmen in elliptischen Kurven
- Die Sicherheit von Curve25519 entspricht 128 Bits eines symmetrischen Schlüssels — im Moment reicht das noch dicke.
- Curve25519 ist effizient implementiert
- Es ist für eine 1:1–Verbindung optimiert
- Jeder Teilnehmer „multipliziert“ seinen geheimen Schlüssel mit dem öffentlichen des Partners, die beiden Produkte

Eingesetzte Technik: Curve25519

- Elliptic Curve Cryptography basiert nicht auf Faktorisierung, sondern auf natürlichen Logarithmen in elliptischen Kurven
- Die Sicherheit von Curve25519 entspricht 128 Bits eines symmetrischen Schlüssels — im Moment reicht das noch dicke.
- Curve25519 ist effizient implementiert
- Es ist für eine 1:1–Verbindung optimiert
- Jeder Teilnehmer „multipliziert“ seinen geheimen Schlüssel mit dem öffentlichen des Partners, die beiden Produkte sind identisch.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität der Sender belegt ist
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität der Sender belegt ist
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität der Sender belegt ist
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität der Sender belegt ist.
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität der Sender bestätigt
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität des Senders belegt ist
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Wurstkessel

Im Moment setze ich ausschließlich Wurstkessel ein, obwohl das Verfahren noch nicht sorgfältig genug geprüft ist:

- Wurstkessel kann Ver/Entschlüsselung und Authentifizierung in einem Durchgang machen, weil am Ende der Verschlüsselung auch ein valider Hash herauskommt, der am Ende der Entschlüsselung ebenfalls verifiziert wird
- Damit löst ein einziger Wurstkesseldurchgang die drei Probleme Vertraulichkeit, Integrität und Authentizität
  1. die Daten sind verschlüsselt
  2. der korrekte Hash belegt ihre Integrität
  3. und die Berechnung des Hashes bedarf, ebenso wie die Verschlüsselung, die Kenntnis des Schlüssels, wodurch die Authentizität des Senders belegt ist
- Man kann das wahrscheinlich auch mit einer geschickten Anwendung von AES machen, aber AES ist nicht dafür konstruiert.

Verdeckte Initialisierungsvektoren

- Der Schlüssels soll nicht wiederverwendet werden (nur für eine 1:1 Retransmission), weil sonst ein Klartextangriff möglich ist
- Übliches Vorgehen: Initialisierungsvektor mit jedem Paket mit übertragen
- Nachteil: Damit wird „der andere Teil“ des Schlüssels bekannt.
- Lösung: Generierung der Initialisierungsvektoren mit einem PRNG (wieder: Wurstkessel) auf beiden Seiten — diese IVs sind „shared secrets“, und nicht bekannt. Nur der Startwert wird übertragen — der geht zusammen mit dem [illegible] (Idee von Helmar Wodke).

Verdeckte Initialisierungsvektoren

- Der Schlüssels soll nicht wiederverwendet werden (nur für eine 1:1 Retransmission), weil sonst ein Klartextangriff möglich ist
- Übliches Vorgehen: Initialisierungsvektor mit jedem Paket mit übertragen
- Nachteil: Damit wird „der andere Teil“ des Schlüssels bekannt.
- Lösung: Generierung der Initialisierungsvektoren mit einem PRNG (wieder: Wurstkessel) auf beiden Seiten — diese IVs sind „shared secrets“, und nicht bekannt. Nur der Startwert wird übertragen — der geht zusammen mit dem [illegible] (Idee von Helmar Wodke).

Verdeckte Initialisierungsvektoren

- Der Schlüssels soll nicht wiederverwendet werden (nur für eine 1:1 Retransmission), weil sonst ein Klartextangriff möglich ist
- Übliches Vorgehen: Initialisierungsvektor mit jedem Paket mit übertragen
- Nachteil: Damit wird „der andere Teil“ des Schlüssels bekannt.
- Lösung: Generierung der Initialisierungsvektoren mit einem PRNG (wieder: Wurstkessel) auf beiden Seiten — diese IVs sind „shared secrets“, und nicht bekannt. Nur der Startwert wird übertragen — der geht zusammen mit dem [illegible] (Idee von Helmar Wodke).

Verdeckte Initialisierungsvektoren

- Der Schlüssels soll nicht wiederverwendet werden (nur für eine 1:1 Retransmission), weil sonst ein Klartextangriff möglich ist
- Übliches Vorgehen: Initialisierungsvektor mit jedem Paket mit übertragen
- Nachteil: Damit wird „der andere Teil“ des Schlüssels bekannt.
- Lösung: Generierung der Initialisierungsvektoren mit einem PRNG (wieder: Wurstkessel) auf beiden Seiten — diese IVs sind „shared secrets“, und nicht bekannt. Nur der Startwert wird übertragen — der geht zusammen mit dem gemeinsamen Schlüssel in die Generierung der IVs ein (Idee von Helmar Wodke).

PKI

Zur Zeit werden drei verschiedene Verfahren benutzt:

1. Hierarchisch mit Certification Authorities (z.B. SSL): Das Vertrauen wird auf „Notare“, also die CAs übertragen, die dann eben vertrauenswürdig sein müssen (alle, da jede CA jedem ein Zertifikat ausstellen kann). Zertifiziert wird der Server, d.h. der Besucher weiß dann, dass er dieser Seite so viel vertrauen kann wie dem übelsten der 600 CAs.
2. Peer to Peer (z.B. PGP): Das Vertrauen wird durch ein „Network of Trust“ erworben, hier muss man entweder selbst nachprüfen, oder man vertraut nur, wenn sich mehrere einig sind
3. Änderungen beobachten (z.B. SSH): Das Vertrauen wird durch wiederholte Kontakte bestärkt, und solange sich nichts ändert, fühlt man sich sicher

PKI

Zur Zeit werden drei verschiedene Verfahren benutzt:

1. Hierarchisch mit Certification Authorities (z.B. SSL): Das Vertrauen wird auf „Notare“, also die CAs übertragen, die dann eben vertrauenswürdig sein müssen (alle, da jede CA jedem ein Zertifikat ausstellen kann). Zertifiziert wird der Server, d.h. der Besucher weiß dann, dass er dieser Seite so viel vertrauen kann wie dem übelsten der 600 CAs.
2. Peer to Peer (z.B. PGP): Das Vertrauen wird durch ein „Network of Trust“ erworben, hier muss man entweder selbst nachprüfen, oder man vertraut nur, wenn sich mehrere einig sind
3. Änderungen beobachten (z.B. SSH): Das Vertrauen wird durch wiederholte Kontakte bestärkt, und solange sich nichts ändert, fühlt man sich sicher

PKI

Zur Zeit werden drei verschiedene Verfahren benutzt:

1. Hierarchisch mit Certification Authorities (z.B. SSL): Das Vertrauen wird auf „Notare“, also die CAs übertragen, die dann eben vertrauenswürdig sein müssen (alle, da jede CA jedem ein Zertifikat ausstellen kann). Zertifiziert wird der Server, d.h. der Besucher weiß dann, dass er dieser Seite so viel vertrauen kann wie dem übelsten der 600 CAs.
2. Peer to Peer (z.B. PGP): Das Vertrauen wird durch ein „Network of Trust“ erworben, hier muss man entweder selbst nachprüfen, oder man vertraut nur, wenn sich mehrere einig sind
3. Änderungen beobachten (z.B. SSH): Das Vertrauen wird durch wiederholte Kontakte bestärkt, und solange sich nichts ändert, fühlt man sich sicher

PKI

Zur Zeit werden drei verschiedene Verfahren benutzt:

1. Hierarchisch mit Certification Authorities (z.B. SSL): Das Vertrauen wird auf „Notare“, also die CAs übertragen, die dann eben vertrauenswürdig sein müssen (alle, da jede CA jedem ein Zertifikat ausstellen kann). Zertifiziert wird der Server, d.h. der Besucher weiß dann, dass er dieser Seite so viel vertrauen kann wie dem übelsten der 600 CAs.
2. Peer to Peer (z.B. PGP): Das Vertrauen wird durch ein „Network of Trust“ erworben, hier muss man entweder selbst nachprüfen, oder man vertraut nur, wenn sich mehrere einig sind
3. Änderungen beobachten (z.B. SSH): Das Vertrauen wird durch wiederholte Kontakte bestärkt, und solange sich nichts ändert, fühlt man sich sicher

Was will man eigentlich überhaupt? 

Die typische Anwendung ist, ein sicheres Login zu bekommen.
Das wirft eine Frage auf:

- Ist es nicht eigentlich der Client, der vertrauenswürdig sein sollte?

Die Verbindung ist für beide hinreichend abgesichert, wenn einer erfolgreich eine Vertrauensprüfung durchgeführt hat.
Dreht man die Vertrauensrelation um, kann man schon mit dem SSH–Ansatz gute Ergebnisse erzielen.

Was will man eigentlich überhaupt? 

Die typische Anwendung ist, ein sicheres Login zu bekommen. Das wirft eine Frage auf:

- Ist es nicht eigentlich der Client, der vertrauenswürdig sein sollte?

Die Verbindung ist für beide hinreichend abgesichert, wenn einer erfolgreich eine Vertrauensprüfung durchgeführt hat. Dreht man die Vertrauensrelation um, kann man schon mit dem SSH–Ansatz gute Ergebnisse erzielen.

Was will man eigentlich überhaupt? 

Die typische Anwendung ist, ein sicheres Login zu bekommen. Das wirft eine Frage auf:

- Ist es nicht eigentlich der Client, der vertrauenswürdig sein sollte?

Die Verbindung ist für beide hinreichend abgesichert, wenn einer erfolgreich eine Vertrauensprüfung durchgeführt hat.
Dreht man die Vertrauensrelation um, kann man schon mit dem SSH–Ansatz gute Ergebnisse erzielen.

Was will man eigentlich überhaupt?

Die typische Anwendung ist, ein sicheres Login zu bekommen. Das wirft eine Frage auf:

- Ist es nicht eigentlich der Client, der vertrauenswürdig sein sollte?

Die Verbindung ist für beide hinreichend abgesichert, wenn einer erfolgreich eine Vertrauensprüfung durchgeführt hat. Dreht man die Vertrauensrelation um, kann man schon mit dem SSH–Ansatz gute Ergebnisse erzielen.

For Further Reading

Bernd Paysan

Fossil Repository und Wiki

`http://fossil.net2o.de/`