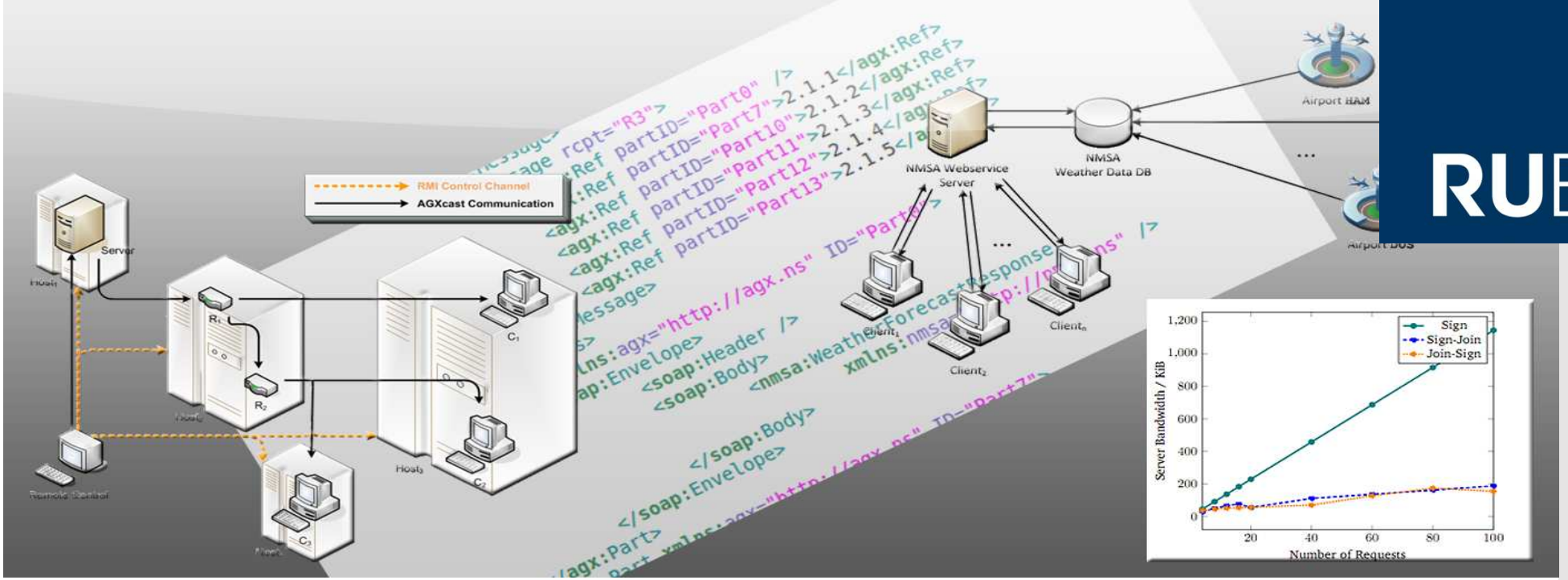


Application and Verification of XML Signatures for Aggregated XML Documents

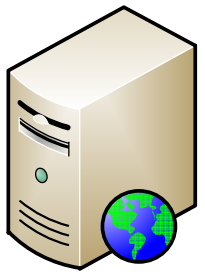
ANDREAS BECKER

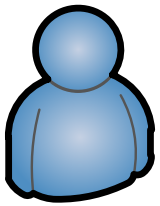
ABSCHLUSSVORTRAG 21.09.2010

BETREUER: DIPL.-INF. MEIKO JENSEN

FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK
Lehrstuhl für Netz- und Datensicherheit
Prof. Dr. Jörg Schwenk

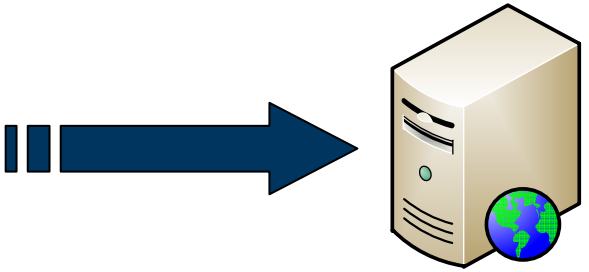
Motivation

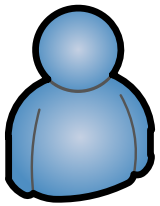




Motivation

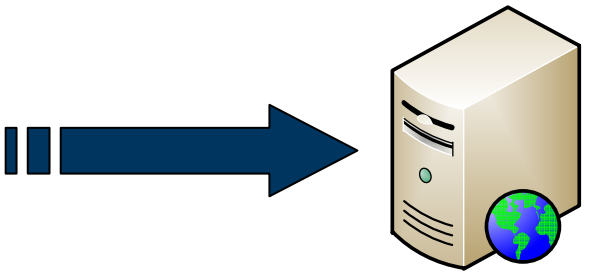
```
<soap:Envelope>
      <soap:Header/>
      <soap:Body>
            <GetTimestampRequest/>
      </soap:Body>
</soap:Envelope>
```

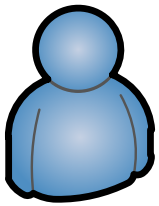




Motivation

```
<soap:Envelope>
      <soap:Header/>
      <soap:Body>
            <GetTimestampRequest/>
      </soap:Body>
</soap:Envelope>
```

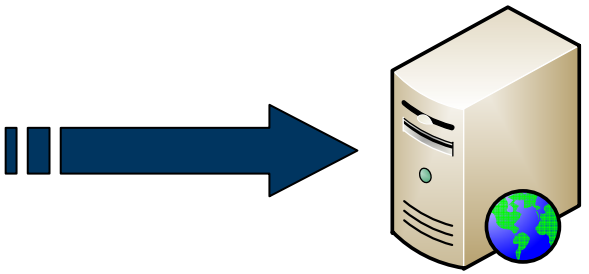


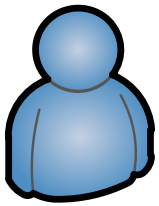


```
<soap:Envelope>
      <soap:Header/>
      <soap:Body>
            <GetTimestampResponse>
                  1283781302
            </GetTimestampResponse>
      </soap:Body>
</soap:Envelope>
```

Motivation

```
<soap:Envelope>
      <soap:Header/>
      <soap:Body>
            <GetTimestampRequest/>
      </soap:Body>
</soap:Envelope>
```





```
<soap:Envelope>
      <soap:Header/>
      <soap:Body>
            <GetTimestampResponse>
                  1283781302
            </GetTimestampResponse>
      </soap:Body>
</soap:Envelope>
```

XML und SOAP (besonders via HTTP) bringen großen Overhead mit!

Motivation (II)

Motivation (II)

- Ansätze zur Einsparung von Bandbreite:
  - Datenkompression
  - IP-Multicast
  - Aggregation *ähnlicher* XML-Dokumente

Motivation (II)

- Ansätze zur Einsparung von Bandbreite:
  - Datenkompression
  - IP-Multicast
  - Aggregation *ähnlicher* XML-Dokumente
- Verschiedene Ansätze zur Aggregation vorhanden

Motivation (II)

- Ansätze zur Einsparung von Bandbreite:
  - Datenkompression
  - IP-Multicast
  - Aggregation *ähnlicher* XML-Dokumente
- Verschiedene Ansätze zur Aggregation vorhanden
- Bislang offene Frage:

Anwendbarkeit digitaler Signaturen gemäß
XML Signature-Standard auf aggregierte Dokumente?

- Untersuchung, basierend auf Arbeit von Azzini et al. [1]

Agenda

RUB

Agenda

1. Einführung

- XML Similarity & Aggregation
- Public Key Systeme & Digitale Signaturen
- XML Signature
- Fallstudie Wetterdienst

RUB

Agenda

1. Einführung

- XML Similarity & Aggregation
- Public Key Systeme & Digitale Signaturen
- XML Signature
- Fallstudie Wetterdienst

2. Aggregated XML

- Aggregation & Wiederherstellung
- Aggregation Signierter Dokumente (**`Sign-Join`**)

  Signieren Aggregierter Dokumente (**`Join-Sign`**)

Agenda (II)

Agenda (II)

3. Xcast & Performance-Messungen

- Einführung Xcast
- Xcast-via-AGX („AGXcast“)
- Framework für Performance-Messungen
- Messergebnisse

Agenda (II)

3. Xcast & Performance-Messungen

- Einführung Xcast
- Xcast-via-AGX („AGXcast“)
- Framework für Performance-Messungen
- Messergebnisse

4. Abschluss

- Zusammenfassung
- Fazit

RUB

Einführung

XML Similarity

XML Similarity

- Ähnlichkeit zweier XML-Dokumente?

XML Similarity

- Ähnlichkeit zweier XML-Dokumente?
  - Einfaches Verfahren: Byteweiser Vergleich
    - Problem: Berücksichtigt Baumstruktur von XML nicht!

XML Similarity

- Ähnlichkeit zweier XML-Dokumente?
  - Einfaches Verfahren: Byteweiser Vergleich
    - Problem: Berücksichtigt Baumstruktur von XML nicht!
  - Phan et al. [2] schlagen zweistufigen Ansatz vor:

$$sim(m_1, m_2) = sim_{temp}(m_1, m_2) \cdot sim_{val}(m_1, m_2)$$

XML Similarity

- Ähnlichkeit zwei
  - Einfaches Verfah
    - Problem: Berüc
  - Phan et al. [2] schlagen z stufigen Ansatz vor:

„Template Similarity“:
Anteil der gemeinsamen Elemente der XML-Dokumente im Verhältnis zu allen auftauchenden Elementnamen.

$$sim(m_1, m_2) = sim_{temp}(m_1, m_2) \cdot sim_{val}(m_1, m_2)$$

XML Similarity

- Ähnlichkeit zweier XML-Dokur
  - Einfaches Verfahren: Byteweise
    - Problem: Berücksichtigt Baumstru
  - Phan et al. [2] schlagen zweistufigen Ah or:

$$sim(m_1, m_2) = sim_{temp}(m_1, m_2) \cdot sim_{val}(m_1, m_2)$$

„Value Similarity“:
Durchschnittliche Ähnlichkeit aller Datenwerte gleichen Typs. Für jeden Datentyp (Boolean, String, Numerisch) separat definiert.

XML Similarity

- Ähnlichkeit zweier XML-Dokumente?
  - Einfaches Verfahren: Byteweiser Vergleich
    - Problem: Berücksichtigt Baumstruktur von XML nicht!
  - Phan et al. [2] schlagen zweistufigen Ansatz vor:

$$\mathrm{sim}(m_1, m_2) = \mathrm{sim}_{\mathrm{temp}}(m_1, m_2) \cdot \mathrm{sim}_{\mathrm{val}}(m_1, m_2)$$

  - Baumstruktur wird *besser* berücksichtigt als oben!

XML Similarity

- Ähnlichkeit zweier XML-Dokumente?
  - Einfaches Verfahren: Byteweiser Vergleich
    - Problem: Berücksichtigt Baumstruktur von XML nicht!
  - Phan et al. [2] schlagen zweistufigen Ansatz vor:

$$\mathrm{sim}(m_1, m_2) = \mathrm{sim}_{temp}(m_1, m_2) \cdot \mathrm{sim}_{val}(m_1, m_2)$$

  - Baumstruktur wird *besser* berücksichtigt als oben!
- Ähnlichkeit $\mathrm{sim}(m_1, m_2)$ ergibt Wert von 0…1

XML Similarity

- Ähnlichkeit zweier XML-Dokumente?
  - Einfaches Verfahren: Byteweiser Vergleich
    - Problem: Berücksichtigt Baumstruktur von XML nicht!
  - Phan et al. [2] schlagen zweistufigen Ansatz vor:

$$\mathrm{sim}(m_1, m_2) = \mathrm{sim}_{temp}(m_1, m_2) \cdot \mathrm{sim}_{val}(m_1, m_2)$$

  - Baumstruktur wird *besser* berücksichtigt als oben!
- Ähnlichkeit $\mathrm{sim}(m_1, m_2)$ ergibt Wert von 0…1
- Dokumente m_1, m_2 heißen **ähnlich**, falls $\mathrm{sim}(m_1, m_2)$ einen Schwellwert überschreitet

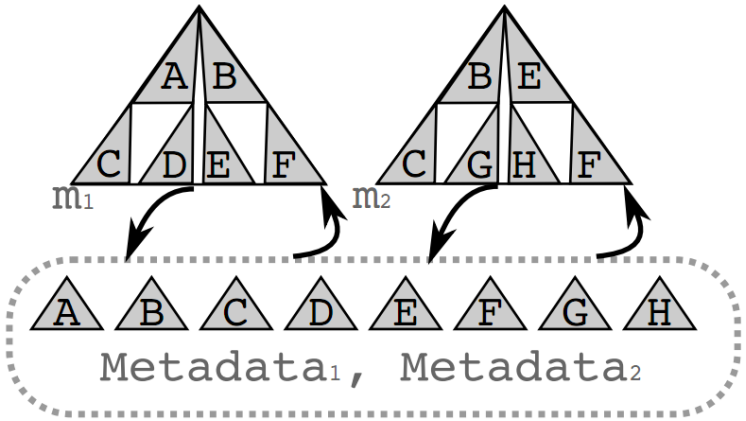
XML Aggregation

XML Aggregation

- Aggregation mehrerer *ähnlicher* XML-Dokumente

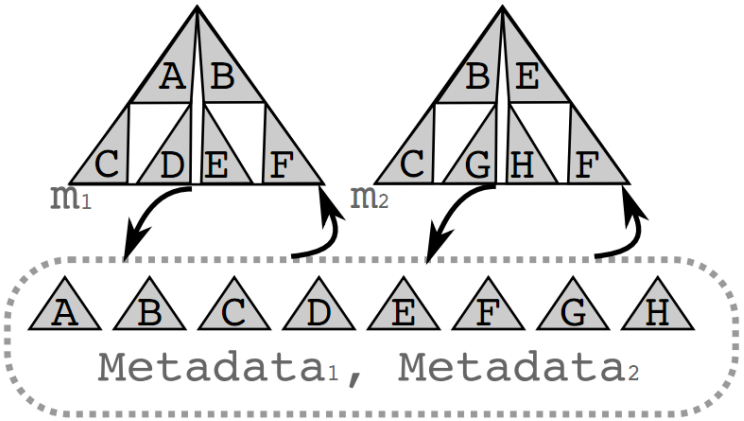
XML Aggregation

- Aggregation mehrerer *ähnlicher* XML-Dokumente
- **Grundidee:**



XML Aggregation

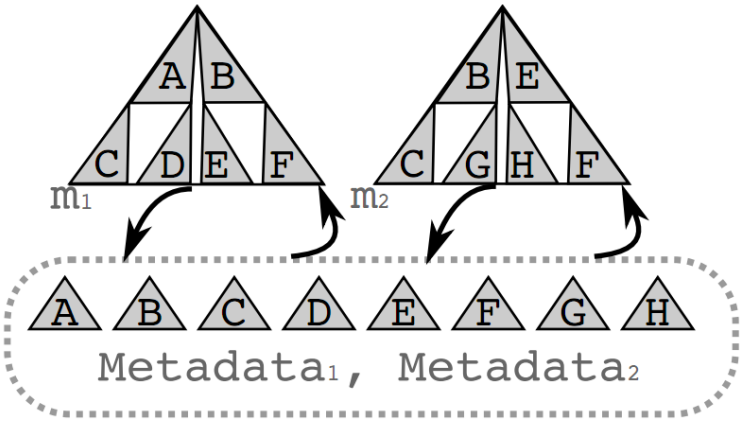
- Aggregation mehrerer *ähnlicher* XML-Dokumente
- **Grundidee:**



- Geschickte Zerlegung der Nachrichten in Teile

XML Aggregation

- Aggregation mehrerer *ähnlicher* XML-Dokumente
- **Grundidee:**



- Geschickte Zerlegung der Nachrichten in Teile
- Reduktion von Redundanz durch Wiederverwendung von Nachrichtenteilen

Public Key-Systeme & Digitale Signaturen

Public Key-Systeme & Digitale Signaturen

- Asymmetrische Krypto: Ein Teilnehmer erzeugt ein Schlüsselpaar (pk, sk) und veröffentlicht pk

Public Key-Systeme & Digitale Signaturen

- Asymmetrische Krypto: Ein Teilnehmer erzeugt ein Schlüsselpaar (pk, sk) und veröffentlicht pk
- Digitale Signatur:
  - Signaturerzeugung: Berechne $s = \mathrm{Sign}_{sk}(m)$
  - Verifikation: $m = \mathrm{Verify}_{pk}(s)$?

Public Key-Systeme & Digitale Signaturen

- Asymmetrische Krypto: Ein Teilnehmer erzeugt ein Schlüsselpaar (pk, sk) und veröffentlicht pk
- Digitale Signatur:
  - Signaturerzeugung: Berechne $s = \mathrm{Sign}_{sk}(m)$
  - Verifikation: $m = \mathrm{Verify}_{pk}(s)$?
- Sicherheitsziele:
  - Authentizität
  - Integrität
  - (Nicht-Zurückweisbarkeit / Verbindlichkeit)

Digitale Signaturen (II)

Digitale Signaturen (II)

- Sicherheit einer Digitalen Signatur:
  - Ein Signaturverfahren heißt **existentiell unfälschbar**, falls kein Angreifer ein gültiges Paar (m,s) erzeugen kann, so dass s = $Sign_{sk}(m)$ gilt, ohne sk zu kennen!

Digitale Signaturen (II)

- Sicherheit einer Digitalen Signatur:
  - Ein Signaturverfahren heißt **existentiell unfälschbar**, falls kein Angreifer ein gültiges Paar (m,s) erzeugen kann, so dass $s = \text{Sign}_{sk}(m)$ gilt, ohne sk zu kennen!
- Hash-and-Sign Paradigma:
  - Um beliebig lange Nachrichten signieren zu können, wird statt m der Hashwert $h(m)$ signiert
  - Empfänger überprüft, ob $h(m) = \text{Verify}_{pk}(s)$ gilt
  - Die Hashfunktion h() muss zusätzlich kollisionsfrei sein, damit das Konstrukt existentiell unfälschbar ist, d.h. ein Angreifer darf kein $m_1 \neq m_2$ finden, so dass $h(m_1) = h(m_2)$

XML Signature

W3C Recommendation [3]

```
<Signature>
      <SignedInfo>
            <CanonicalizationMethod/>
            <SignatureMethod/>
            (<Reference URI=„…“>
                  <Transforms>
                        (<Transform/>)*
                  </Transforms>
                  <DigestMethod/>
                  <DigestValue/>
            </Reference>)+
      </SignedInfo>
      <SignatureValue/>
      (<KeyInfo/>)?
      (<Object/>)?
</Signature>
```

X
W3C

„SignedInfo“:
Dieser gesamte Block wird kanonisiert, gehashed und signiert. Das Ergebnis **s** wird im Element **<SignatureValue>** platziert.

→ **s** = Sign(h(c14n(**<SignedInfo>**)))

```
<Signature>
      <SignedInfo>
            <CanonicalizationMethod/>
            <SignatureMethod/>
            (<Reference URI=„…“>
                  <Transforms>
                        (<Transform/>)*
                  </Transforms>
                  <DigestMethod/>
                  <DigestValue/>
            </Reference>)+
      </SignedInfo>
      <SignatureValue/>
      (<KeyInfo/>)?
      (<Object/>)?
</Signature>
```

XML Signature

W3C Recommendation [3]

```
<Signature>
      <SignedInfo>
            <CanonicalizationMethod/>
            <SignatureMethod/>
            (<Reference URI=„…“>
                  <Transforms>
                        (<Transform/>)*
                  </Transforms>
                  <DigestMethod/>
```

„References“:

Jede **<Reference>** zeigt auf eine Ressource, deren URI dereferenziert und transformiert wird. Der Hashwert des Ergebnisses hiervon steht im jeweiligen **<DigestValue>**.

→ $\mathbf{DigVal_i} = h_i(\text{Transforms}_i(\text{Deref}(\mathbf{URI_i})))$

Jede Ressource wird einer **individuellen** Liste von Transformationen, sowie einer **individuellen** Hashfunktion unterworfen!

XML Signature (II)

XML Signature (II)

Erinnerung: $s = Sign_{sk}(h(c14n(\ldots \mathbf{DigVal_i})))$

XML Signature (II)

Erinnerung: $s = Sign_{sk}(h(c14n(\ldots \mathbf{DigVal_i})))$

- Verifikation der XML Signature:
  - Reference Validation: Überprüfe für alle References, ob $\mathbf{DigVal_i} = h_i(Transforms_i(Deref(\mathbf{URI_i})))$ gilt
  - Signature Validation: Überprüfe, ob $Verify_{pk}(s) = h(c14n(\mathbf{<SignedInfo>}))$ gilt

XML Signature (II)

Erinnerung: $s = Sign_{sk}(h(c14n(\ldots \mathbf{DigVal_i})))$

- Verifikation der XML Signature:
  - Reference Validation: Überprüfe für alle References, ob $\mathbf{DigVal_i} = h_i(Transforms_i(Deref(\mathbf{URI_i})))$ gilt
  - Signature Validation: Überprüfe, ob $Verify_{pk}(s) = h(c14n(\mathbf{<SignedInfo>}))$ gilt
- Signatur ist gültig ↔ Beide Schritte erfolgreich

Sicherheit der XML Signature

Sicherheit der XML Signature

RUB

- Existentiell unfälschbar, wenn Sign() existentiell unfälschbar und h **und alle h_i** kollisionsfrei sind

Sicherheit der XML Signature

- Existentiell unfälschbar, wenn Sign() existentiell unfälschbar und h **und alle h_i** kollisionsfrei sind
- Besonderheit: Jede Reference ist einzeln prüfbar!

Sicherheit der XML Signature

- Existentiell unfälschbar, wenn Sign() existentiell unfälschbar und h **und alle h_i** kollisionsfrei sind
- Besonderheit: Jede Reference ist einzeln prüfbar!
- Großes Problem der XML Signature:

$$\mathbf{DigVal_i} = h_i(\ \mathrm{Transforms}_i(\ \mathrm{Deref}(\ \mathbf{URI_i}\ )\ )\ )$$

Sicherheit der XML Signature

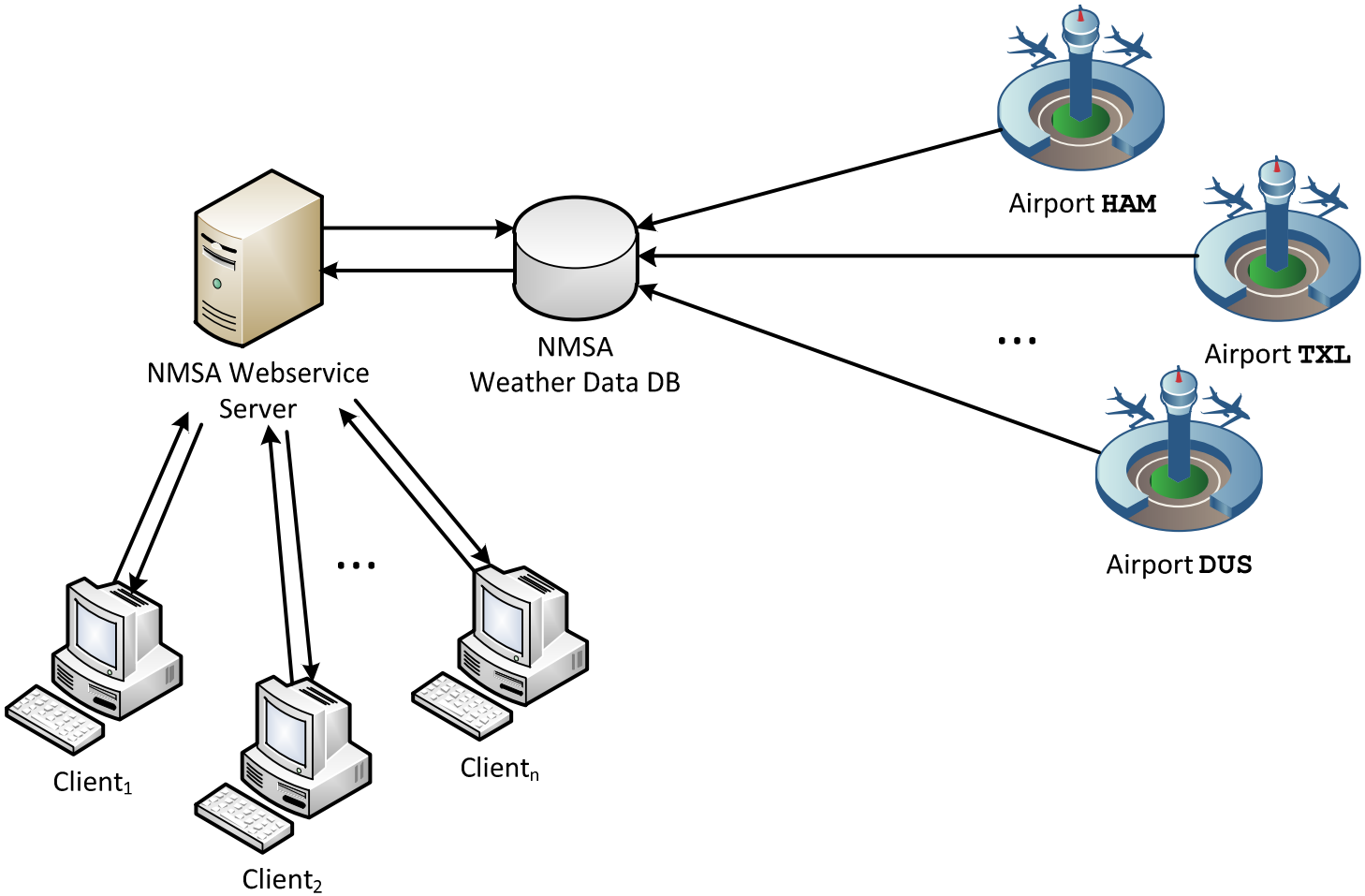
- Existentiell unfälschbar, wenn Sign() existentiell unfälschbar und h **und alle h_i** kollisionsfrei sind
- Besonderheit: Jede Reference ist einzeln prüfbar!
- Großes Problem der XML Signature:

$$\mathbf{DigVal_i} = h_i(\ \mathrm{Transforms}_i(\ \mathrm{Deref}(\ \mathbf{URI_i}\ )\ )\ )$$

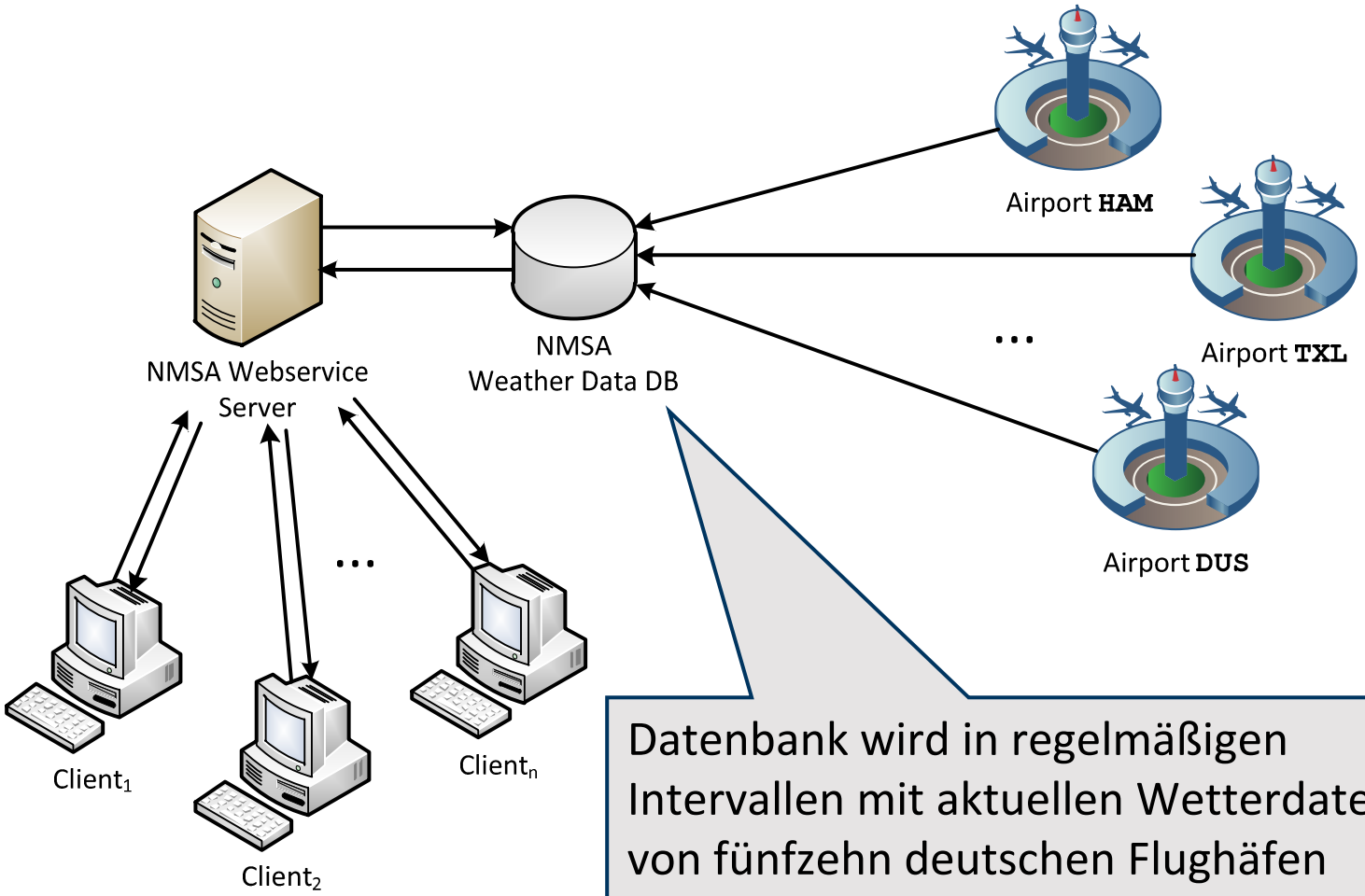
URI Resolving, XPath-Filter, XSLT-Transformationen, …

→ **What you see is what you signed?**

Fallstudie Wetterdienst



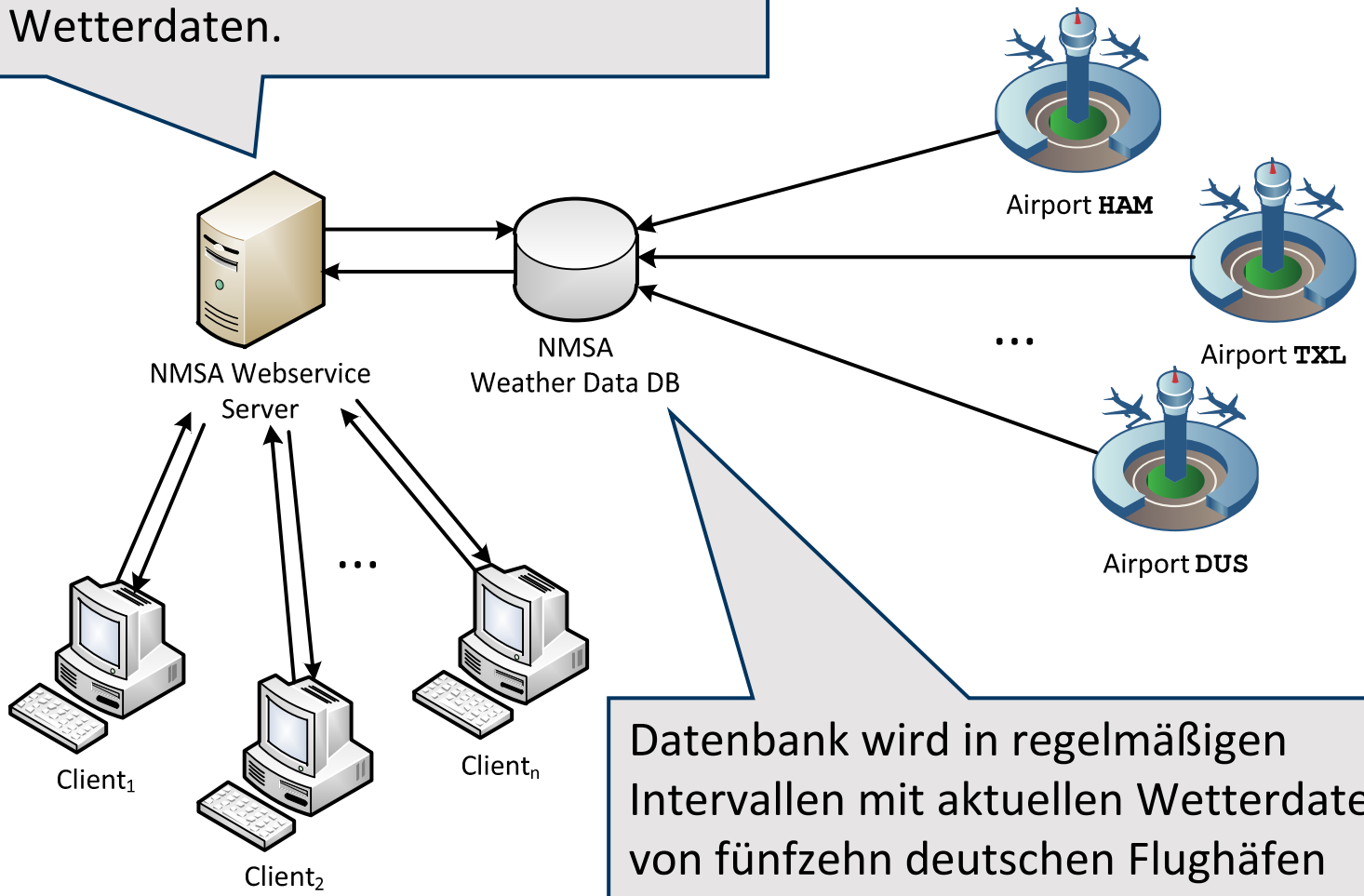
Fallstudie Wetterdienst



Datenbank wird in regelmäßigen Intervallen mit aktuellen Wetterdaten von fünfzehn deutschen Flughäfen aktualisiert.

Fallstudie Wetterdienst

SOAP-basierter Webservice zur Anfrage von Wetterdaten.



Datenbank wird in regelmäßigen Intervallen mit aktuellen Wetterdaten von fünfzehn deutschen Flughäfen aktualisiert.

Fallstudie Wetterdienst (II)

RUB

Fallstudie Wetterdienst (II)

- Clients fragen Wetterdaten für Flughäfen an

Fallstudie Wetterdienst (II)

- Clients fragen Wetterdaten für Flughäfen an
- Aktuelles Wetter oder Vorhersage für 3h oder 24h

Fallstudie Wetterdienst (II)

- Clients fragen Wetterdaten für Flughäfen an
- Aktuelles Wetter oder Vorhersage für 3h oder 24h

Fallstudie Wetterdienst (II)

RUB

- Clients fragen Wetterdaten für Flughäfen an
- Aktuelles Wetter oder Vorhersage für 3h oder 24h

```
…
<NMSAForecastRequest>
      <WeatherData airportCode=„HAM“ time=„current“/>
      <WeatherData airportCode=„MUC“ time=„24h“/>
</NMSAForecastRequest>
…
```

Fallstudie Wetterdienst (II)

- Clients fragen Wetterdaten für Flughäfen an
- Aktuelles Wetter oder Vorhersage für 3h oder 24h

```
…
<NMSAForecastRequest>
      <WeatherData airportCode=„HAM“ time=„current“/>
      <WeatherData airportCode=„MUC“ time=„24h“/>
</NMSAForecastRequest>
…
```

- Server sendet die gleichen **<WeatherData>**-Elemente zurück, jeweils um eine Baumstruktur erweitert, mit aktuellen Daten aus der Datenbank

RUB

Aggregated XML

Motivation Aggregated XML

Motivation Aggregated XML

RUB

- Verschiedene Datenformate für XML Aggregation

Motivation Aggregated XML

RUB

- Verschiedene Datenformate für XML Aggregation
- Häufig wird ein *Common Part* vorausgesetzt

Motivation Aggregated XML

- Verschiedene Datenformate für XML Aggregation
- Häufig wird ein *Common Part* vorausgesetzt
- Hier: Neues, flexibles Datenformat
  „Aggregated XML“ (AGX)

Motivation Aggregated XML

RUB

- Verschiedene Datenformate für XML Aggregation
- Häufig wird ein *Common Part* vorausgesetzt
- Hier: Neues, flexibles Datenformat
  „Aggregated XML“ (AGX)
- Setzt die Kernidee der Aggregation (s.o.) um, ohne weitere Annahmen über Eingabedokumente

Motivation Aggregated XML

RUB

- Verschiedene Datenformate für XML Aggregation
- Häufig wird ein *Common Part* vorausgesetzt
- Hier: Neues, flexibles Datenformat
  „Aggregated XML“ (AGX)
- Setzt die Kernidee der Aggregation (s.o.) um, ohne weitere Annahmen über Eingabedokumente
- Idee: Ist XML Signature auf solche Dokumente anwendbar, dann auch auf aggregierte Dokumente beliebiger anderer (performanterer) Formate

Aggregation am Beispiel

```
<A>
    <B>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

```
<A>
    <B>
        <C>Some Text</C>
        <E>
            <F/>
        </E>
    </B>
</A>
```

Aggregation am Beispiel

RUB

```
<A>
    <B>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

```
<A>
    <B>
        <C>Some Text</C>
        <E>
            <F/>
        </E>
    </B>
</A>
```

Aggregation

```
<agx:Part ID="Part1">
    <A>
        <B>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID="Part2">
    <D/>
</agx:Part>

<agx:Part ID="Part3">
    <E>
        <F/>
    </E>
</agx:Part>
```

Aggregation am Beispiel

```
<A>
    <B>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

```
<A>
        <E>
            <F/>
        </E>
    </B>
</A>
```

Part:
Entspricht einem Nachrichtenteil, der über seine ID referenziert werden kann. Durch Wiederverwendung solcher Teile kann Redundanz verringert werden.

```
<agx:Part ID="Part1">
    <A>
        <B>
            <C>Some Text</C>
        </B>
    </A>

D="Part2">

<agx:Part ID="Part3">
    <E>
        <F/>
    </E>
</agx:Part>
```

Aggregation am Beispiel

```
<agx:Messages>

<agx:Message>
    <agx:Ref partID="Part1"/>
    <agx:Ref partID="Part2">1.2
       </agx:Ref>
</agx:Message>

<agx:Message>
    <agx:Ref partID="Part1"/>
    <agx:Ref partID="Part3">1.2
       </agx:Ref>
<agx:Message>

</agx:Messages>
```

```
<agx:Part ID="Part1">
    <A>
        <B>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID="Part2">
    <D/>
</agx:Part>

<agx:Part ID="Part3">
    <E>
        <F/>
    </E>
</agx:Part>
```

Aggregation am Beispiel

```
<agx:Messages>

<agx:Message>
    <agx:Ref par ID="Part1"/>
    <agx:Ref part rt2">1.2
      </agx:Ref>
</agx:Message>

<agx:Message>
    <agx:Ref pa
    <agx:Ref partID="Part3">1.2
      </agx:Ref>
<agx:Message>

</agx:Messages>
```

```
<agx:Part ID="Part1">
    <A>
        <B>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID="Part3">
    <E>
        <F/>
    </E>
</agx:Part>
```

Message:
Enthält die Metadaten, um eine individuelle Nachricht aus den Teilen wieder zurück zu gewinnen.

Aggregation am Beispiel

RUB

```
<agx:Messages>

<agx:Message>
    <agx:Ref partID="Part1"/>
    <agx:Ref partID="Part2">1.2
      </agx:Ref>
</agx:Message>

<agx:Message>
    <agx:Ref partI
    <agx:Ref partI
      </agx:Ref>
<agx:Message>

</agx:Messages>
```

```
<agx:Part ID="Part1">
    <A>
        <B>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>
```

Ref:
Hängt den referenzierten Teil an der angegebenen Stelle im Nachrichtenbaum ein:

- **<Ref/>** referenziert das Wurzelelement
- **<Ref>1.2</Ref>** hängt das Element als zweites Kind des ersten Kindes des Wurzelelements ein

Annahme: Refs sind nach ihren „Location Strings“ sortiert. Dies sollte clientseitig sichergestellt werden!

Aggregation Signierter Dokumente (**`Sign-Join`**)

Aggregation Signierter Dokumente (**`Sign-Join`**)

- Kombination von XML Signature und Aggregation:
  - Signiere Einzelnachrichten $m_1, \ldots, m_n \rightarrow m^*_1, \ldots, m^*_n$
  - Aggregiere signierte Nachrichten $m^*_1, \ldots, m^*_n \rightarrow m_{\text{Sign-Join}}$

Aggregation Signierter Dokumente (**Sign-Join**)

- Kombination von XML Signature und Aggregation:
  - Signiere Einzelnachrichten $m_1, \ldots, m_n \rightarrow m^*_1, \ldots, m^*_n$
  - Aggregiere signierte Nachrichten $m^*_1, \ldots, m^*_n \rightarrow m_{\text{Sign-Join}}$
- Es müssen n Einzelsignaturen angewandt werden

Aggregation Signierter Dokumente (**`Sign-Join`**)

- Kombination von XML Signature und Aggregation:
  - Signiere Einzelnachrichten $m_1, \ldots, m_n \rightarrow m^*_1, \ldots, m^*_n$
  - Aggregiere signierte Nachrichten $m^*_1, \ldots, m^*_n \rightarrow m_{\text{Sign-Join}}$
- Es müssen n Einzelsignaturen angewandt werden
- Möglichkeit zusätzlicher Reduktion von Redundanz:
  - Große Teile des **`<Signature>`**-Elementes
  - Base64-kodierte Zertifikate und Schlüsselmaterial

Aggregation Signierter Dokumente (**`Sign-Join`**)

- Kombination von XML Signature und Aggregation:
  - Signiere Einzelnachrichten $m_1, \ldots, m_n \rightarrow m^*_1, \ldots, m^*_n$
  - Aggregiere signierte Nachrichten $m^*_1, \ldots, m^*_n \rightarrow m_{\text{Sign-Join}}$
- Es müssen n Einzelsignaturen angewandt werden
- Möglichkeit zusätzlicher Reduktion von Redundanz:
  - Große Teile des **`<Signature>`**-Elementes
  - Base64-kodierte Zertifikate und Schlüsselmaterial
- Verwendung von Standard XML-Signatures
  - Gleiches Sicherheitsniveau wie XML-Signature (s.o.)!

Signieren Aggregierter Dokumente (**`Join-Sign`**)

Signieren Aggregierter Dokumente (**`Join-Sign`**)

- Intuition:
  - Einzelnachrichten werden aggregiert, *einmal* signiert
  - Unterwegs werden unbenötigte (ggf. signierte) Teile verworfen
    → Signatur wird ungültig?!

Signieren Aggregierter Dokumente (**`Join-Sign`**)

- Intuition:
  - Einzelnachrichten werden aggregiert, *einmal* signiert
  - Unterwegs werden unbenötigte (ggf. signierte) Teile verworfen
    → Signatur wird ungültig?!
- Angepasste Reference Validation beim Empfänger!
  - Wenn eine Reference ins Leere zeigt, wird diese für das Ergebnis der Reference Validation ignoriert
  - Einzelhashes aller References bleiben im Dokument, sodass Signature Validation weiterhin funktioniert!

Join-Sign: Verarbeitung beim Sender

RUB

- Geg.: Signing Policy als XPath-Filter **`/A[1]/B[1]`**

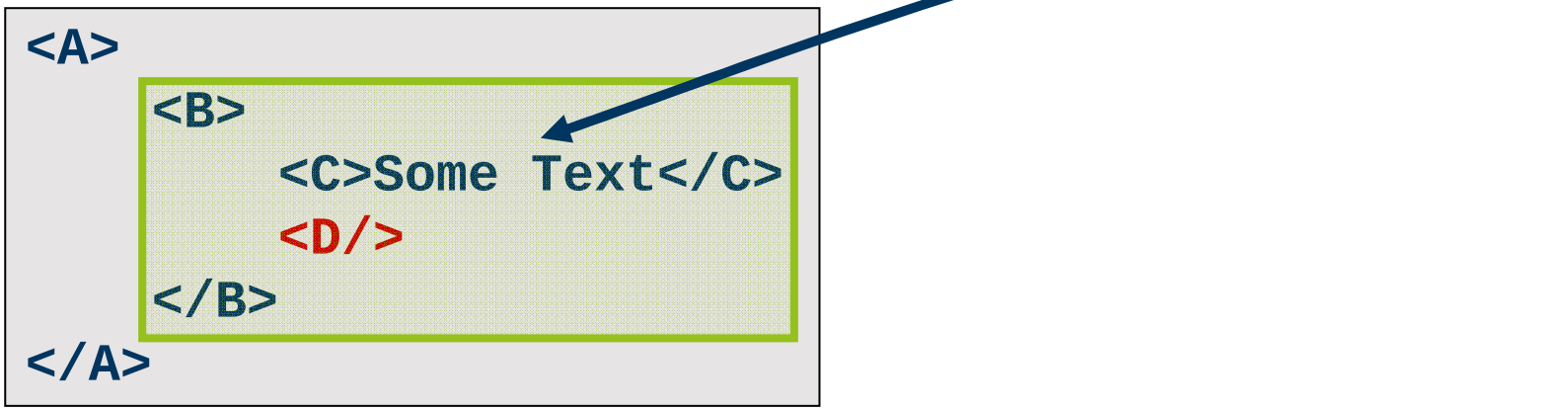
Join-Sign: Verarbeitung beim Sender

- Geg.: Signing Policy als XPath-Filter **`/A[1]/B[1]`**

```
<A>
    <B>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

Join-Sign: Verarbeitung beim Sender

- Geg.: Signing Policy als XPath-Filter **`/A[1]/B[1]`**



Join-Sign: Verarbeitung beim Sender

- Geg.: Signing Policy als XPath-Filter **/A[1]/B[1]**

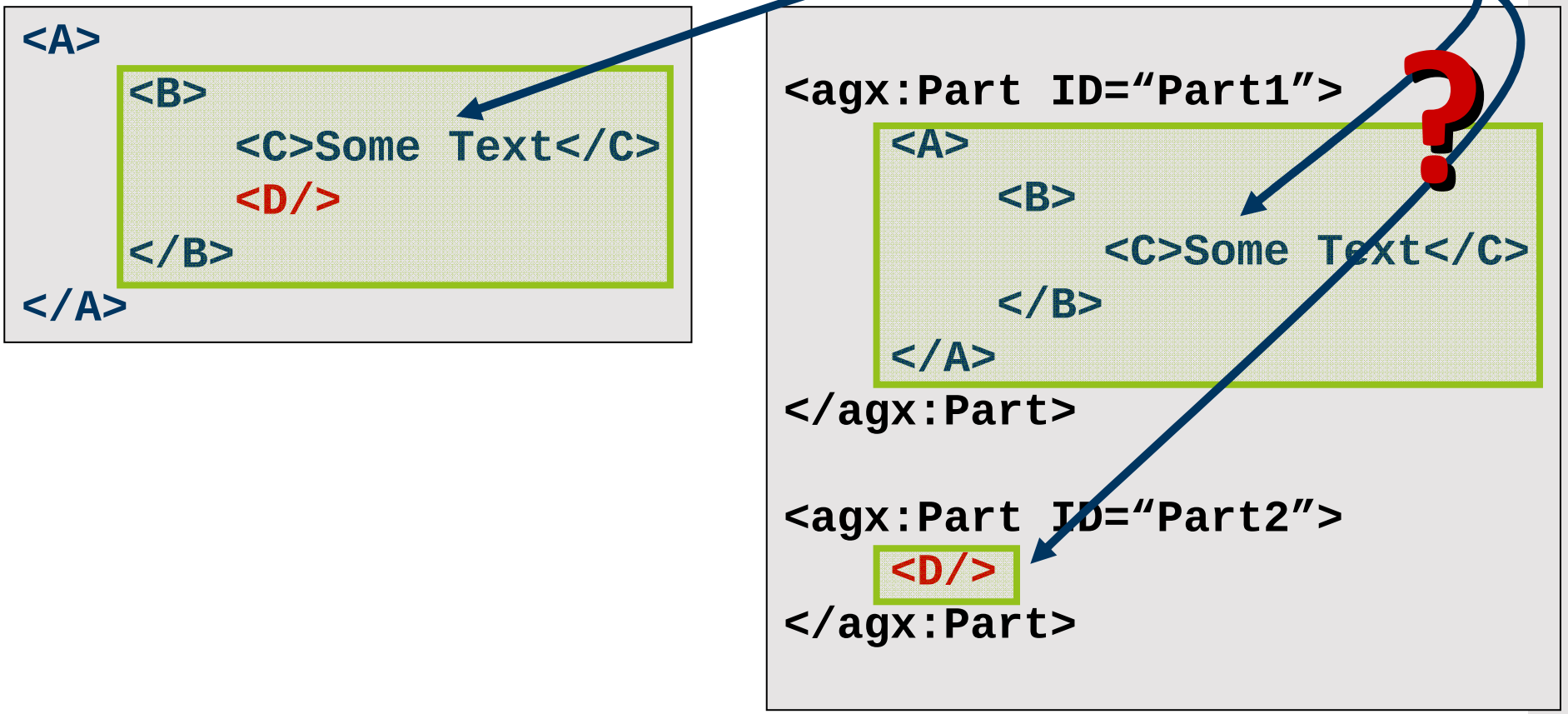
```
<A>
    <B>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

```
<agx:Part ID="Part1">
    <A>
        <B>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID="Part2">
    <D/>
</agx:Part>
```

Join-Sign: Verarbeitung beim Sender

- Geg.: Signing Policy als XPath-Filter **/A[1]/B[1]**



`Join-Sign`:
Verarbeitung beim Sender (II)

Join-Sign: Verarbeitung beim Sender (II)

- Übersetzung des XPath-Ausdrucks?

Join-Sign: Verarbeitung beim Sender (II)

- Übersetzung des XPath-Ausdrucks?
- Probleme:
  - Auslagerung zu signierender Teilbäume in eigene Parts
  - Teile, die nur an *manche* Empfänger signiert sein sollen
  - Auch ID-basierte Referenzierung funktioniert nicht mehr, da IDs im Aggregatdokument nicht eindeutig!

Join-Sign: Verarbeitung beim Sender (II)

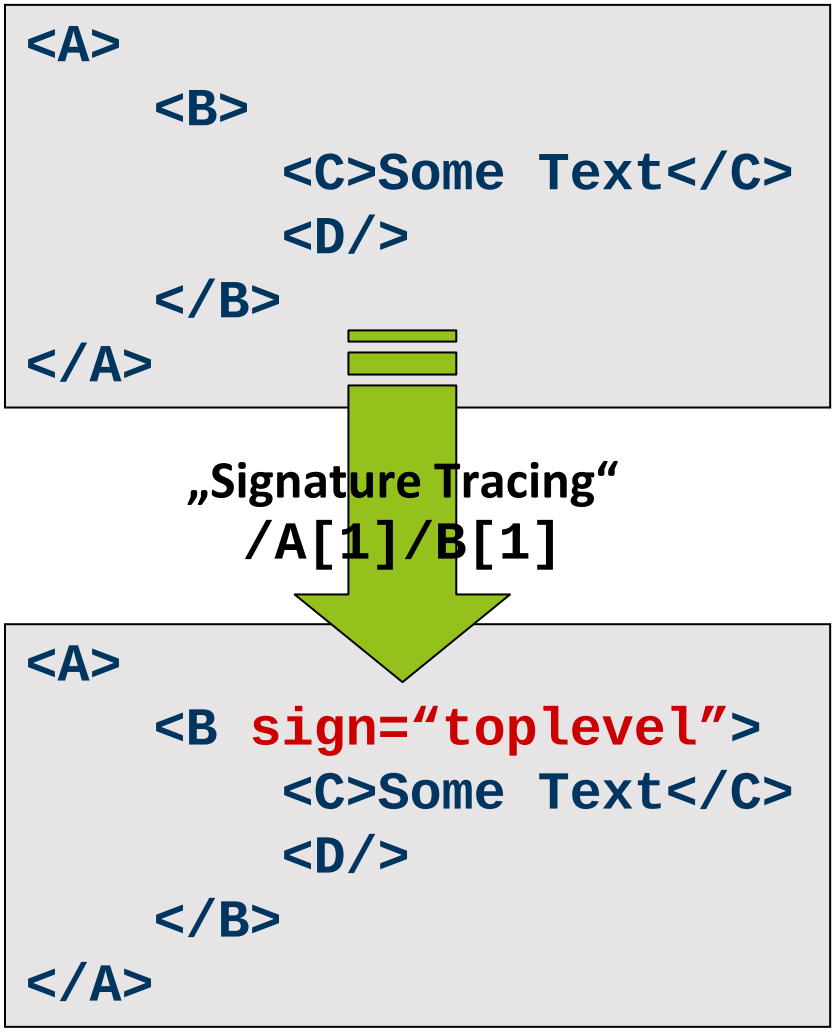
- Übersetzung des XPath-Ausdrucks?
- Probleme:
  - Auslagerung zu signierender Teilbäume in eigene Parts
  - Teile, die nur an *manche* Empfänger signiert sein sollen
  - Auch ID-basierte Referenzierung funktioniert nicht mehr, da IDs im Aggregatdokument nicht eindeutig!
- Lösung „**Trace-Join-Sign**“:
  - XPath-Filter auf Einzeldokumente anwenden
  - Zu signierende Teilbäume markieren
  - Aggregation
- Bei Auslagerung eines Teilbaums: Markierung *erweitern*

„Trace-Join-Sign“

am Beispiel

```
<A>
    <B>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

„Trace-Join-Sign“
am Beispiel



„Trace-Join-Sign“ am Beispiel

```
<A>
    <B sign=“toplevel”>
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

Aggregation

+

Erweiterung der Markierungen

```
<agx:Part ID=“Part1”>
    <A>
        <B sign=“toplevel”>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID=“Part2” sign=“true”>
    <D/>
</agx:Part>
```

„Trace-Join-Sign“ am Beispiel

Bei der Aggregation wurde dieser Part mit **sign=true** markiert, weil ein Vorfahre von **<D/>** im Originaldokument mit **sign=toplevel** markiert ist.

```
<A>
    <B sign="toplevel">
        <C>Some Text</C>
        <D/>
    </B>
</A>
```

```
<agx:Part ID="Part1">
    <A>
        <B sign="toplevel">
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID="Part2" sign="true">
    <D/>
</agx:Part>
```

Aggregation

+

Erweiterung der Markierungen

„Trace-Join-Sign“
am Beispiel

```
<agx:Part ID=“Part1”>
    <A/>
</agx:Part>

<agx:Part ID=“Part2”>
    <D/>
</agx:Part>

<agx:Part ID=“Part3”>
    <B>
        <C>Some Text</C>
    </B>
</agx:Part>
```

Auslagern „primär“ referenzierter Parts

```
<agx:Part ID=“Part1”>
    <A>
        <B sign=“toplevel”>
            <C>Some Text</C>
        </B>
    </A>
</agx:Part>

<agx:Part ID=“Part2” sign=“true”>
    <D/>
</agx:Part>
```

„Trace-Join-Sign“
am Beispiel

RUB

```
<agx:Part ID=“Part1”>
    <A/>
</agx:Part>

<agx:Part ID=“Part2”>
    <D/>
</agx:Part>

<agx:Part ID=“Part3”>
    <B>
        <C>Some Text</C>
    </B>
</agx:Part>
```

XML Signature:

```
<Reference URI=„#Part2“/>
<Reference URI=„#Part3“/>
<Reference>
  <XPath>…/Messages[1]</XPath>
</Reference>
```

„Trace-Join-Sign“
am Beispiel

```
<agx:Part ID=“Part1”>
    <A/>
</agx:Part>

<agx:Part ID=“Part2”>
    <D/>
</agx:Part>

<agx:Part ID=“Part3”>
    <B>
        <C>Some Text</C>
    </B>
</agx:Part>
```

XML Signature:

```
<Reference URI=„#Part2“/>
<Reference URI=„#Part3“/>
<Reference>
  <XPath>…/Messages[1]</XPath>
</Reference>
```

Message-Element:

```
<Ref partID=„Part1“/>
<Ref partID=„Part3“>1</Ref>
<Ref partID=„Part2“>1.2</Ref>
```

Sicherheit der „**Join-Sign** Signature“

Sicherheit der „**Join-Sign** Signature“

- Intuitiv:
  - Message-Element (Steuerung der Applikationslogik) und XML Signature verwenden gleiche Referenzierung
  - Angreifer kann…
    - keine eigenen Teile an signierte Parts anhängen
    - keine (signierten) Parts entfernen
    - die Komposition der Einzelnachricht nicht manipulieren

Sicherheit der „**Join-Sign** Signature“

- Intuitiv:
  - Message-Element (Steuerung der Applikationslogik) und XML Signature verwenden gleiche Referenzierung
  - Angreifer kann…
    - keine eigenen Teile an signier[illegible]nhängen
    - keine (signierten) Parts entfe[illegible]
    - die Komposition de[illegible]

Annahme:
Beide Dereferenzierungs-Module für ID-basierte Referenzierung liefern für eine gegebene ID jeweils das gleiche Element!

Teilweise problematisch bei mehrfach vergebenen IDs, oder IDs verschiedener Namespaces...

Sicherheit der „**Join-Sign** Signature“

RUB

- Intuitiv:
  - Message-Element (Steuerung der Applikationslogik) und XML Signature verwenden gleiche Referenzierung
  - Angreifer kann…
    - keine eigenen Teile an signierte Parts anhängen
    - keine (signierten) Parts entfernen
    - die Komposition der Einzelnachricht nicht manipulieren
- Etwas formaler:
  - Es gibt eine Funktion Retrieve(), die aus den signierten Teilen P^* und **<Messages>** die signierten Teilbäume der Nachricht eindeutig rekonstruiert
  - Beide Parameter dieser Funktion sind signiert
  - Somit bietet die Join-Sign Signature die Sicherheit der XML Sig.

RUB

Xcast & Performance-Messungen

Das Xcast-Protokoll

Das Xcast-Protokoll

- E**x**plicit Multi-uni**cast** Protocol (exp., RFC 5058 [4])

Das Xcast-Protokoll

- E**x**plicit Multi-uni**cast** Protocol (exp., RFC 5058 [4])
- Versand **einer** Nachricht an **n** Empfänger

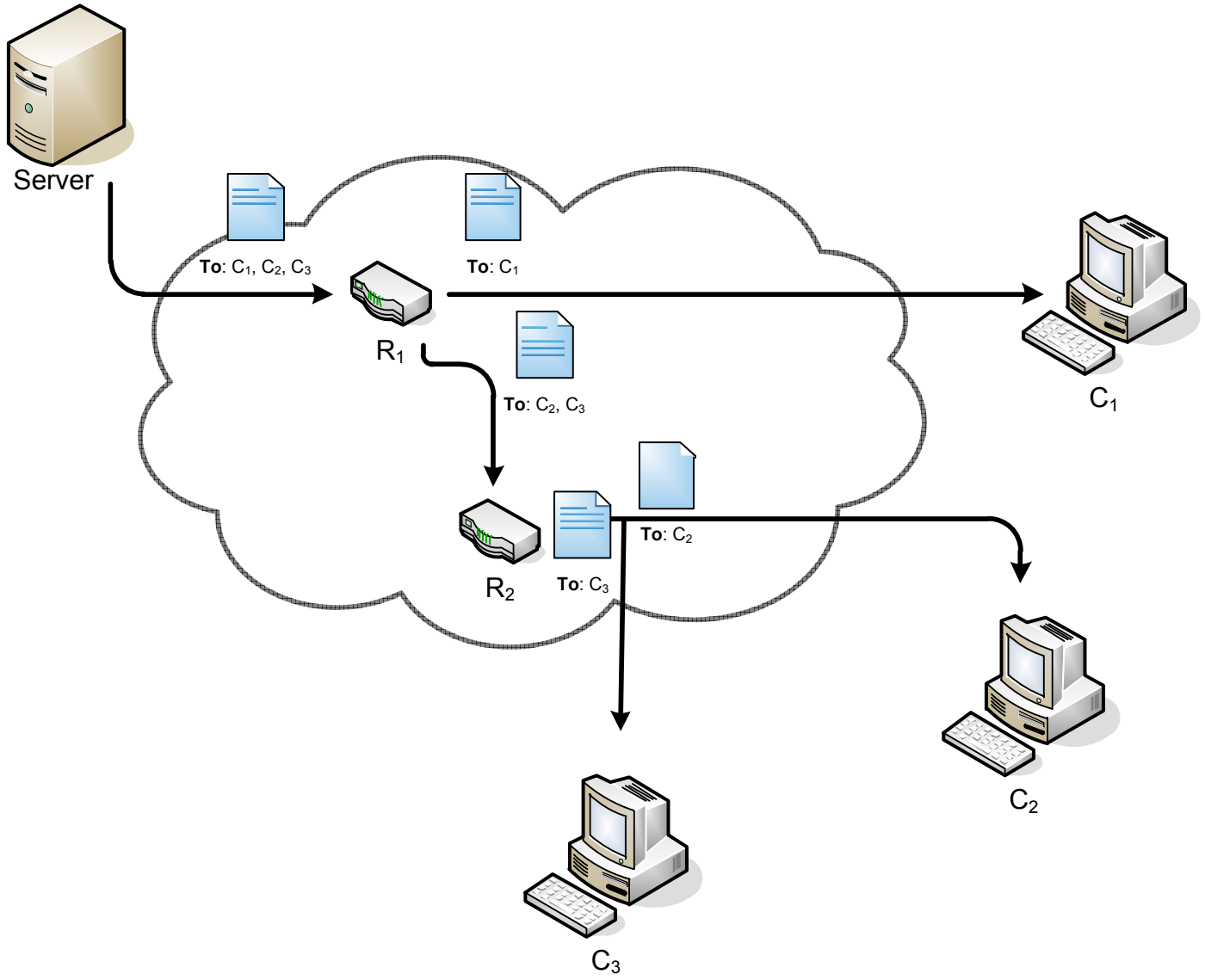
Das Xcast-Protokoll

- E**x**plicit Multi-uni**cast** Protocol (exp., RFC 5058 [4])
- Versand **einer** Nachricht an **n** Empfänger
- Probleme von IP-Multicast:
  - Große Anzahl kleiner Multicastgruppen

Das Xcast-Protokoll

- E**x**plicit Multi-uni**cast** Protocol (exp., RFC 5058 [4])
- Versand **einer** Nachricht an **n** Empfänger
- Probleme von IP-Multicast:
  - Große Anzahl kleiner Multicastgruppen
- Lösung beim Xcast-Protokoll:
  - Empfängeradressen explizit im Header angegeben
  - Router ermitteln next hop für jeden Empfänger mit existierenden Routingprotokollen und senden Kopie der Nachricht an diesen Router

Xcast am Beispiel



Similarity-based SOAP Multicasting Protocol (SMP)

RUB

Similarity-based SOAP Multicasting Protocol (SMP)

- Phan et al. verwenden Grundidee von Xcast für aggregierte SOAP-Nachrichten → SMP

Similarity-based SOAP Multicasting Protocol (SMP)

- Phan et al. verwenden Grundidee von Xcast für aggregierte SOAP-Nachrichten → SMP
- Server versendet aggregierte Nachrichten ähnlicher SOAP-Responses an verschiedene Empfänger

Similarity-based SOAP Multicasting Protocol (SMP)

- Phan et al. verwenden Grundidee von Xcast für aggregierte SOAP-Nachrichten → SMP
- Server versendet aggregierte Nachrichten ähnlicher SOAP-Responses an verschiedene Empfänger
- Aggregatnachrichten werden periodisch versandt

Similarity-based SOAP Multicasting Protocol (SMP)

- Phan et al. verwenden Grundidee von Xcast für aggregierte SOAP-Nachrichten → SMP
- Server versendet aggregierte Nachrichten ähnlicher SOAP-Responses an verschiedene Empfänger
- Aggregatnachrichten werden periodisch versandt
- Empfängeradressen werden ebenfalls im SMP-Header platziert

Similarity-based SOAP Multicasting Protocol (SMP)

- Phan et al. verwenden Grundidee von Xcast für aggregierte SOAP-Nachrichten → SMP
- Server versendet aggregierte Nachrichten ähnlicher SOAP-Responses an verschiedene Empfänger
- Aggregatnachrichten werden periodisch versandt
- Empfängeradressen werden ebenfalls im SMP-Header platziert
- Router verwerfen nicht benötigte Nachrichtenteile

AGXcast

AGXcast

- AGXcast := „Xcast mit AGX-Dokumenten“

AGXcast

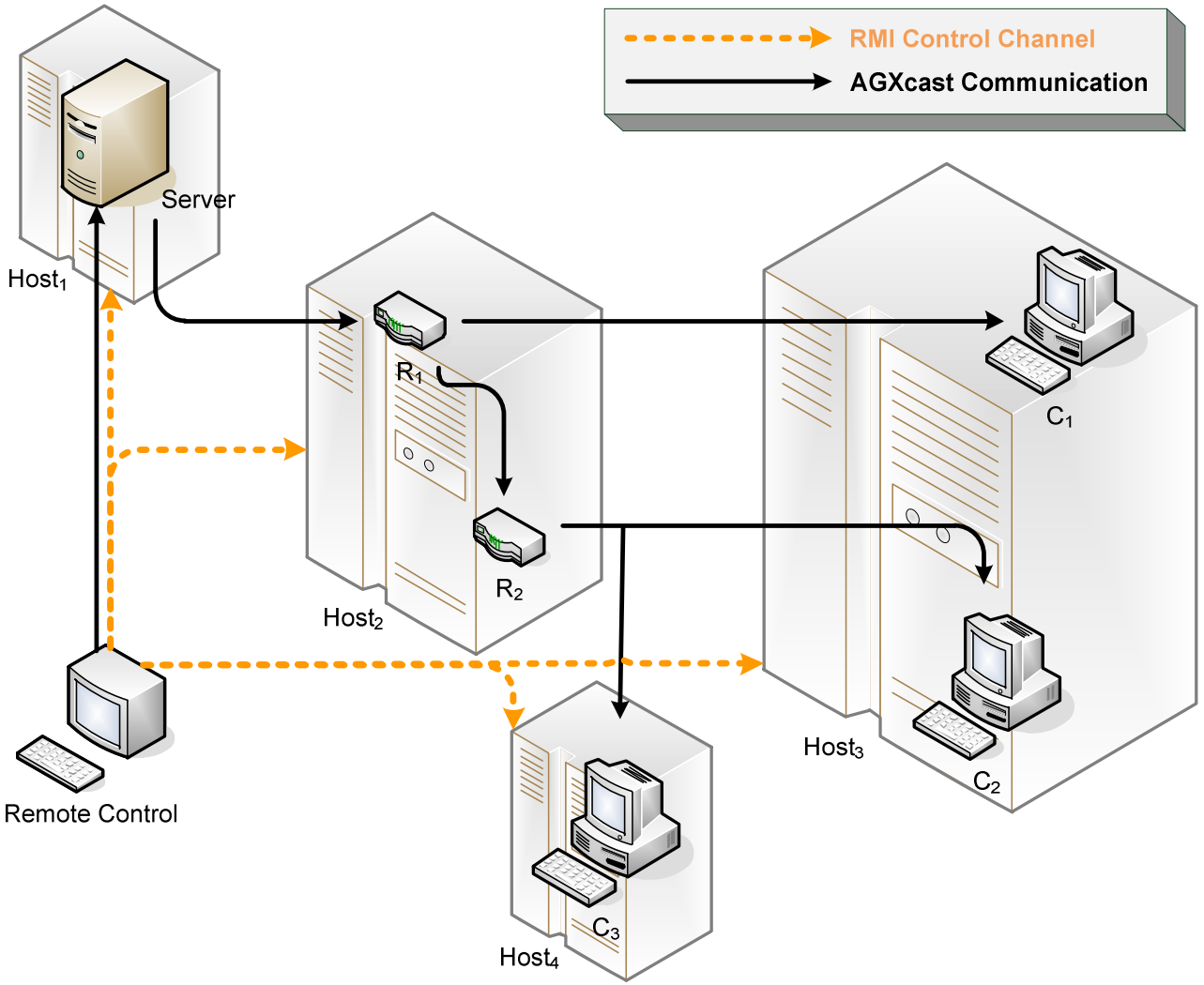
- AGXcast := „Xcast mit AGX-Dokumenten“
- Anders als Phan et al.: Statisches Routing

AGXcast

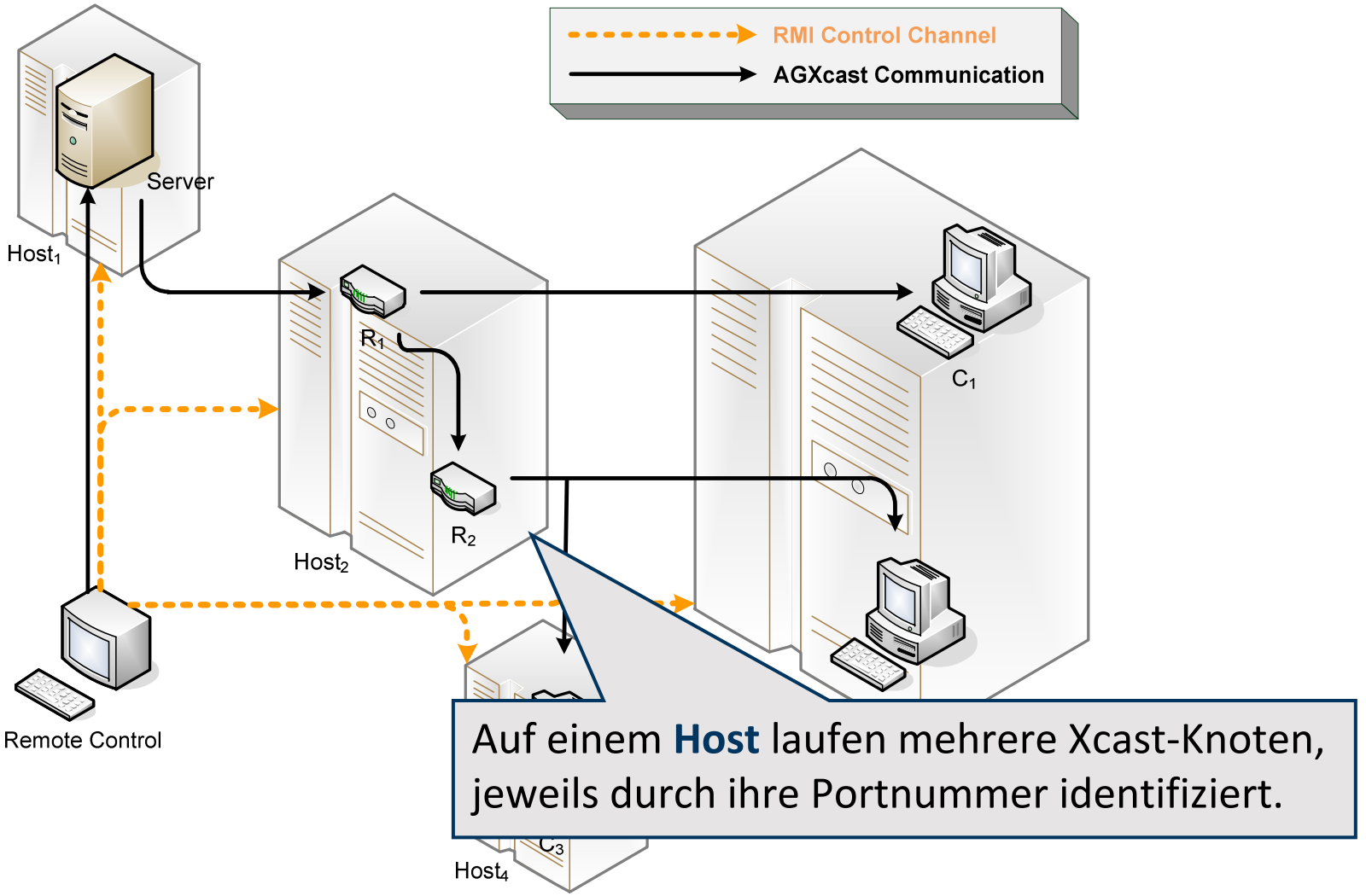
- AGXcast := „Xcast mit AGX-Dokumenten“
- Anders als Phan et al.: Statisches Routing
- DOM-basierte Java-Implementierung von…
  - AGX.Common – Kernfunktionalität für AGX
    - Domainspezifische Aggregatoren
    - Restliche Klassen „generisch“
  - AGXcast – Server-, Router- und Client-Logik
    - Server versendet aggregierte NMSA-Responses

  Framework zur Performance-Messung

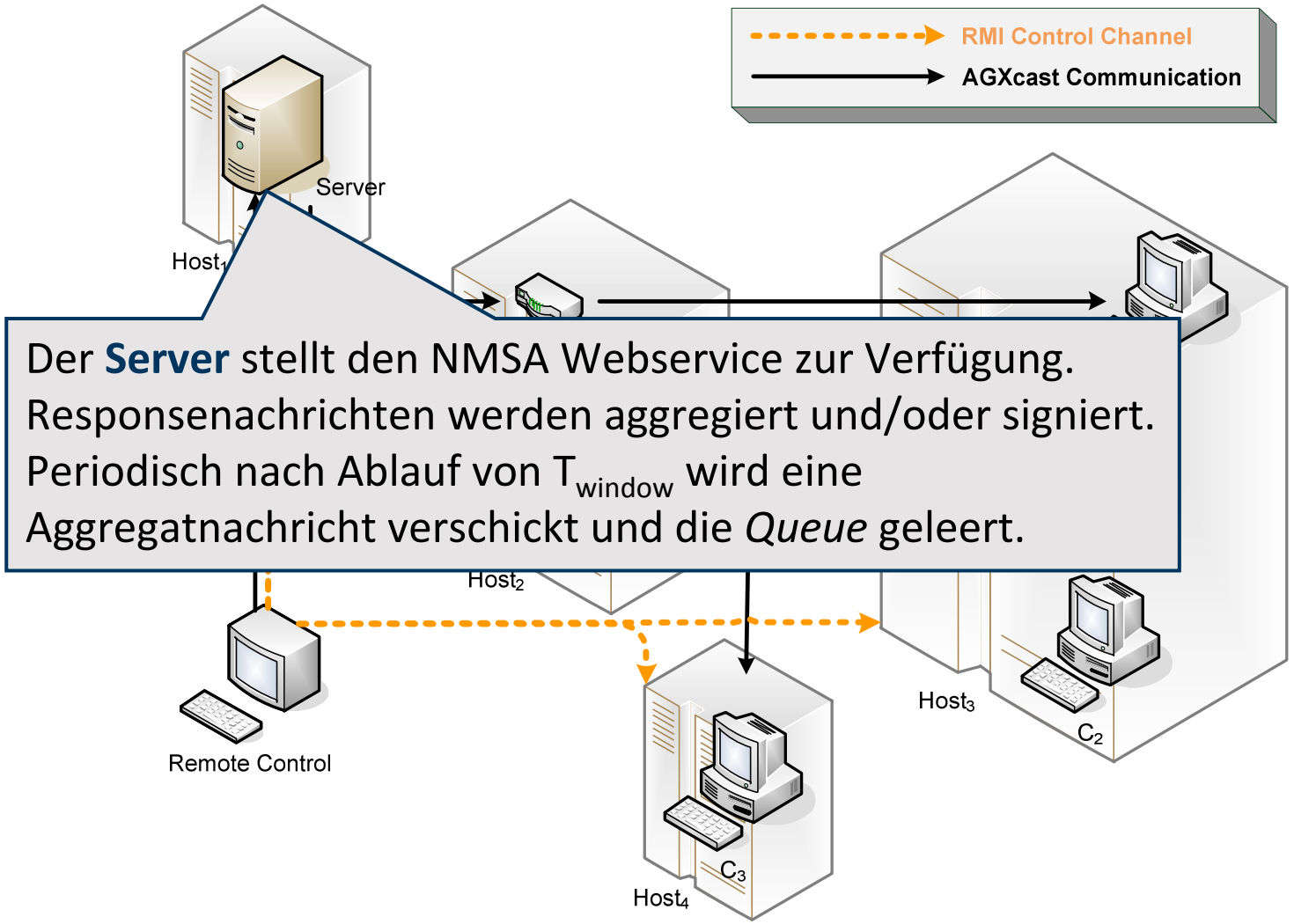
Framework für Performance-Messungen



Framework für Performance-Messungen

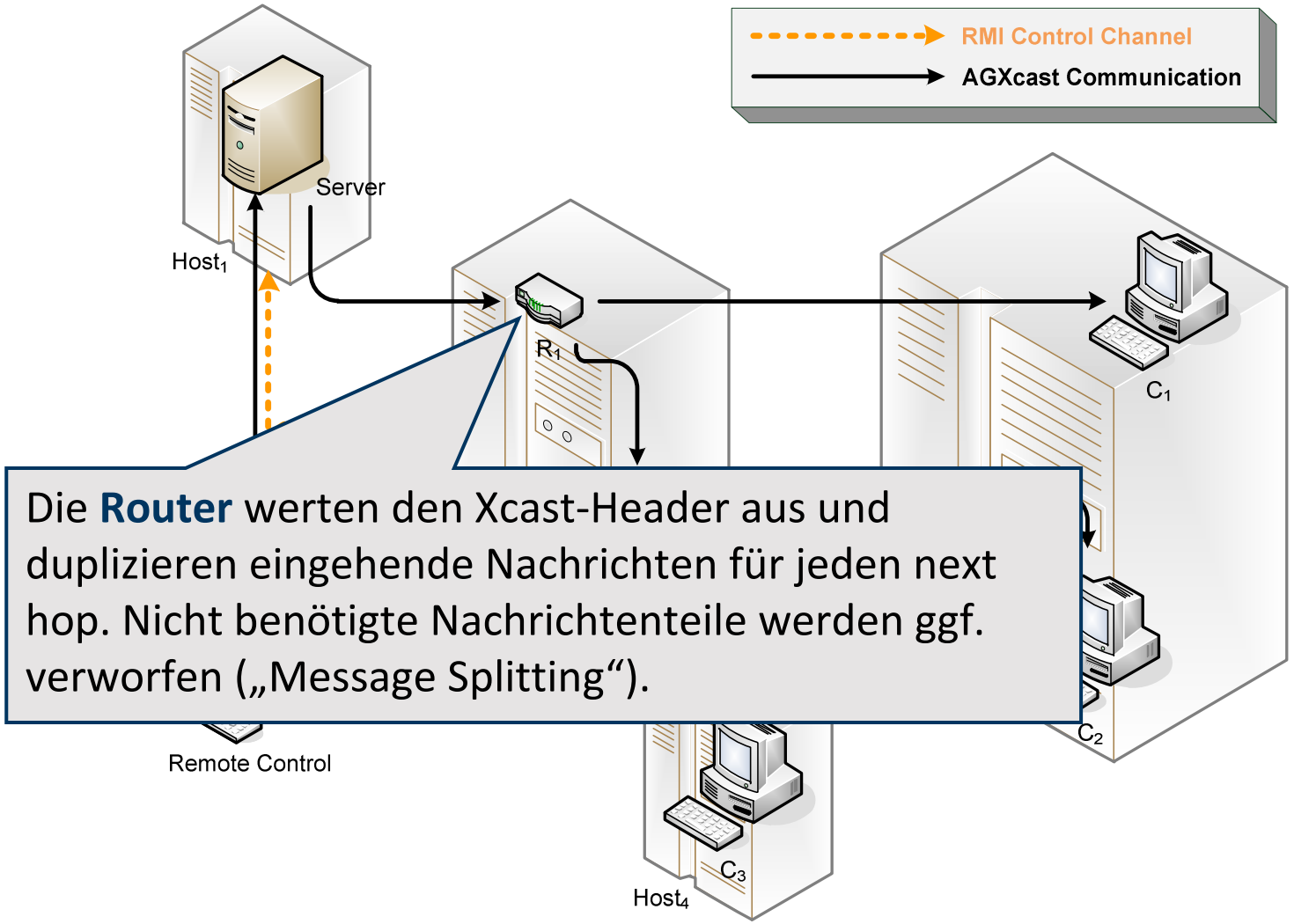


Framework für Performance-Messungen



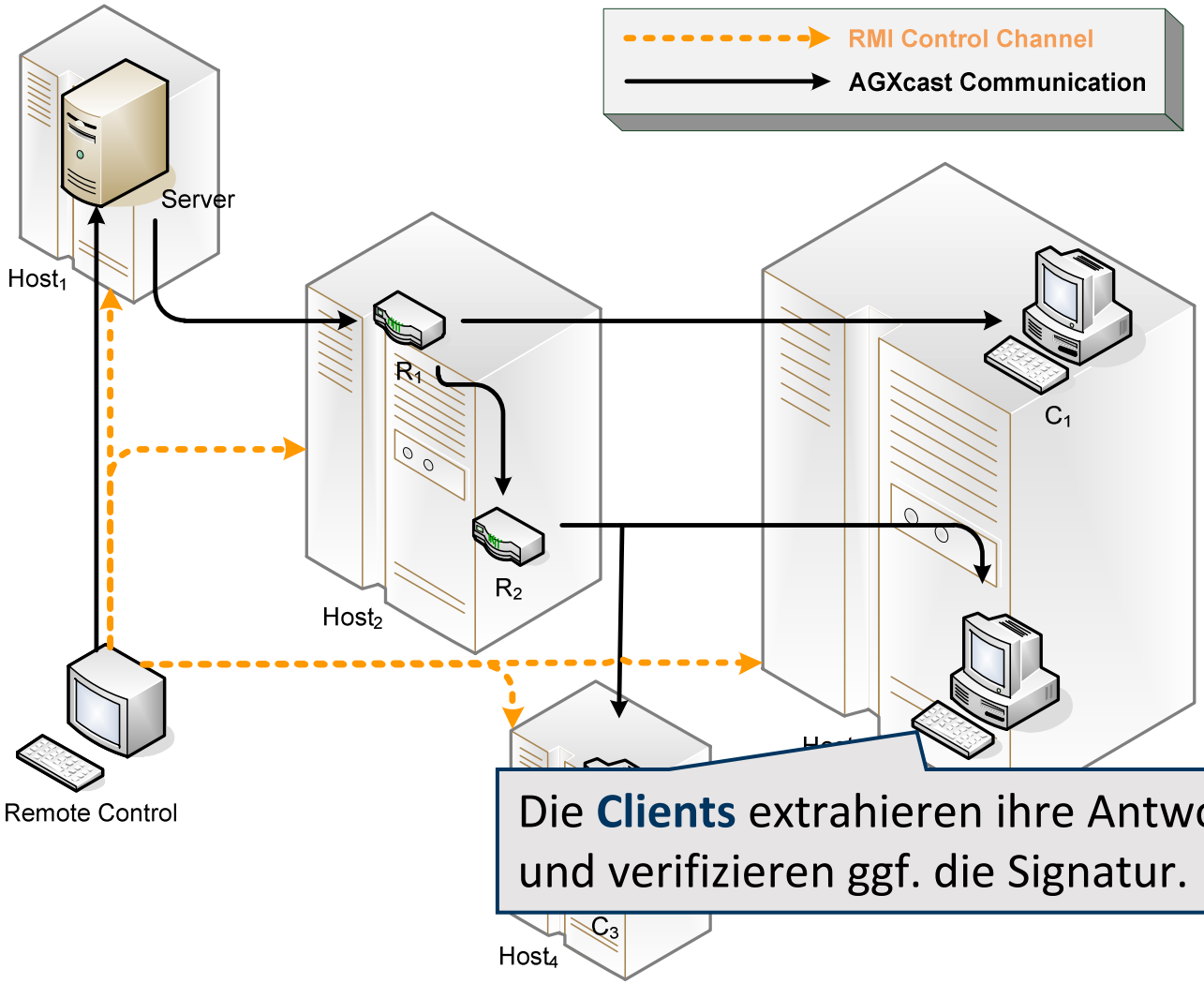
Der **Server** stellt den NMSA Webservice zur Verfügung.
Responsenachrichten werden aggregiert und/oder signiert.
Periodisch nach Ablauf von T_{window} wird eine Aggregatnachricht verschickt und die *Queue* geleert.

Framework für Performance-Messungen



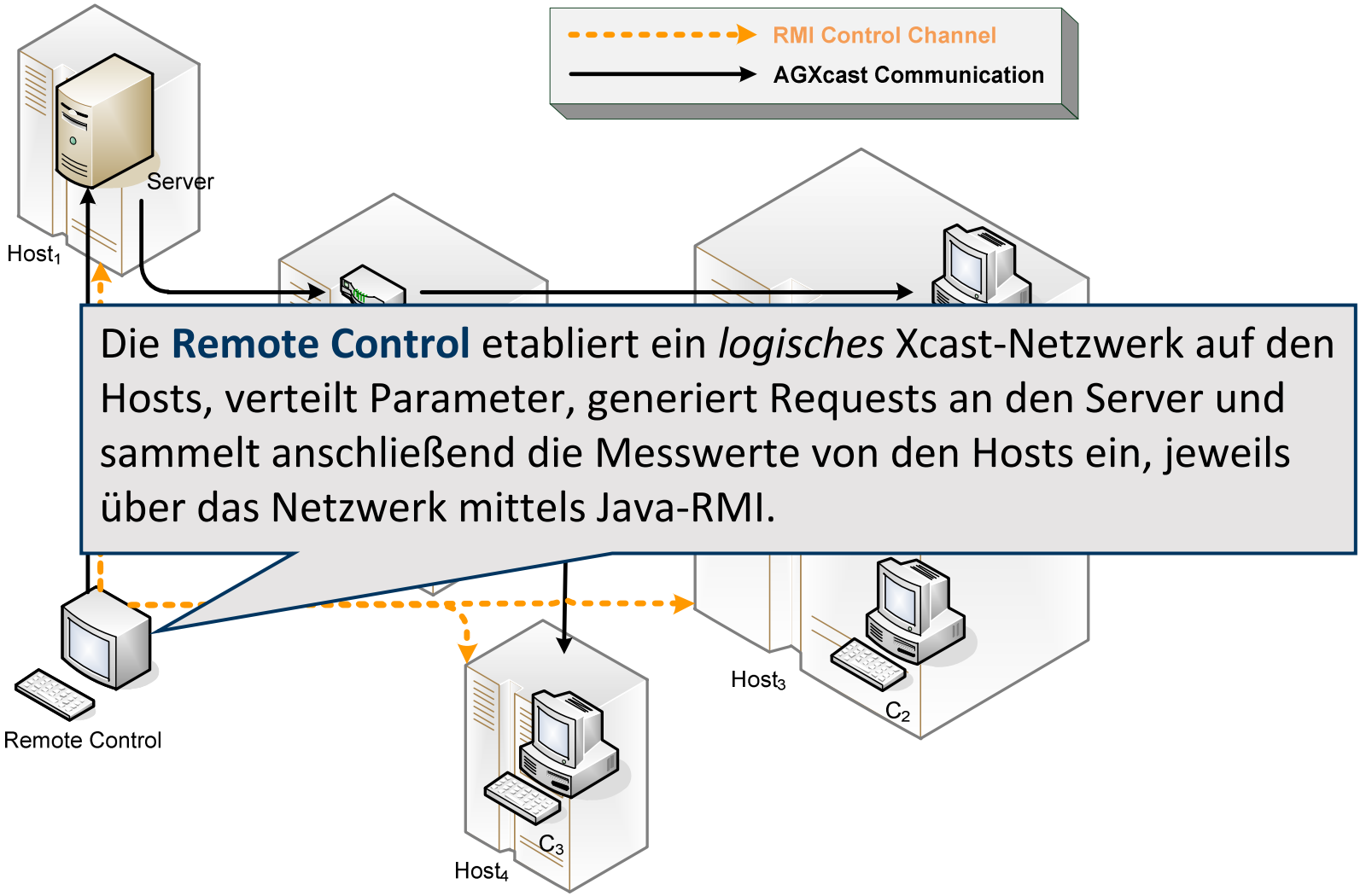
Die **Router** werten den Xcast-Header aus und duplizieren eingehende Nachrichten für jeden next hop. Nicht benötigte Nachrichtenteile werden ggf. verworfen („Message Splitting“).

Framework für Performance-Messungen



Die **Clients** extrahieren ihre Antwortnachrichten und verifizieren ggf. die Signatur.

Framework für Performance-Messungen



Die **Remote Control** etabliert ein *logisches* Xcast-Netzwerk auf den Hosts, verteilt Parameter, generiert Requests an den Server und sammelt anschließend die Messwerte von den Hosts ein, jeweils über das Netzwerk mittels Java-RMI.

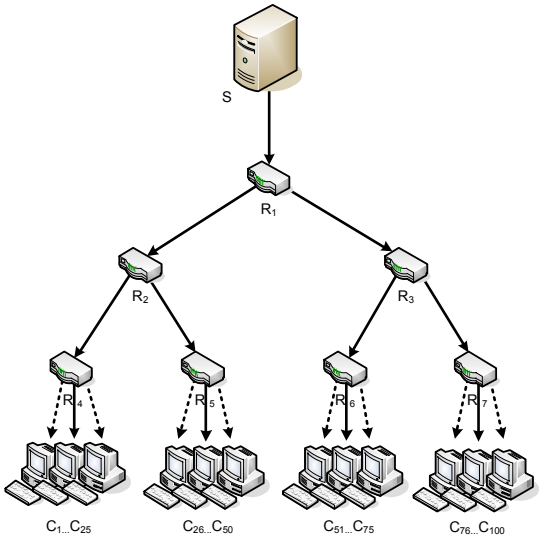
Parameter

Name	Wert
Signing Policy	`/soap:Envelope[1]/soap:Body[1]`
Queueing Interval	1000ms
Datenbasis	Kurze Textdaten, keine großen Binärdaten
# angefragter Tupel / Request	20
Request-Strategie	***variabel*** (Pseudozufällig, 0% / 100% redundant)
Netzwerk-Topologie	***variabel*** (Binärbaum, Kette)
Use Case	***variabel*** (1 oder mehr Requests pro Client)
Verarbeitungsschritte	***variabel*** (SIGN | SIGN-JOIN | JOIN-SIGN)
# Requests / Messung	***variabel*** → x-Achse

Parameter

Sogenannte **sekundäre Parameter** blieben konstant in den Messungen. Auswirkungen dieser Parameter auf die Messwerte werden in der Ausarbeitung diskutiert!

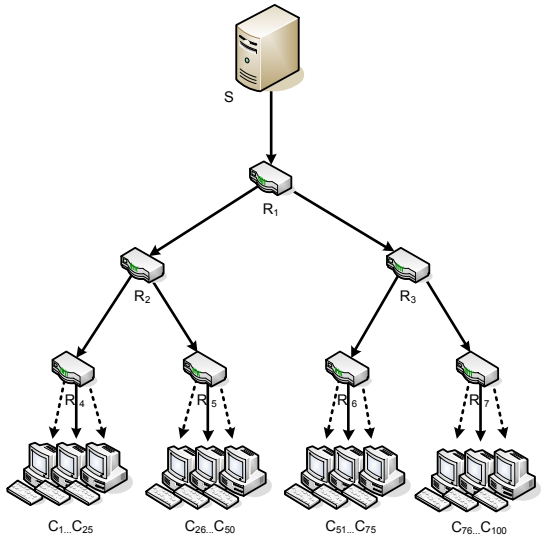
Name	
Signing Policy	**[illegible]**
Queueing Interval	1000ms
Datenbasis	Kurze Textdaten, keine großen Binärdaten
# angefragter Tupel / Request	20
Request-Strategie	***variabel*** (Pseudozufällig, 0% / 100% redundant)
Netzwerk-Topologie	***variabel*** (Binärbaum, Kette)
Use Case	***variabel*** (1 oder mehr Requests pro Client)
Verarbeitungsschritte	***variabel*** (SIGN | SIGN-JOIN | JOIN-SIGN)
# Requests / Messung	***variabel*** → x-Achse



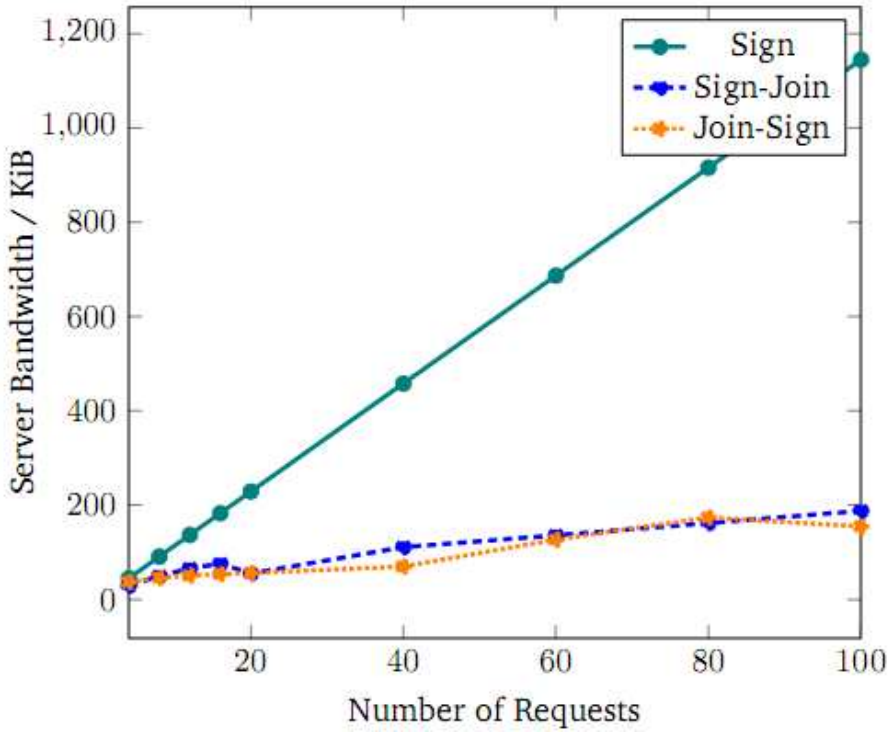
Messergebnisse Binärbaum

- Jeweils 25 Clients mit R_4…R_7 verbunden
- Jeweils ein Request pro Client
- Requests bzw. Responses werden gemäß „Round-Robin“ auf Clients von R_4…R_7 verteilt

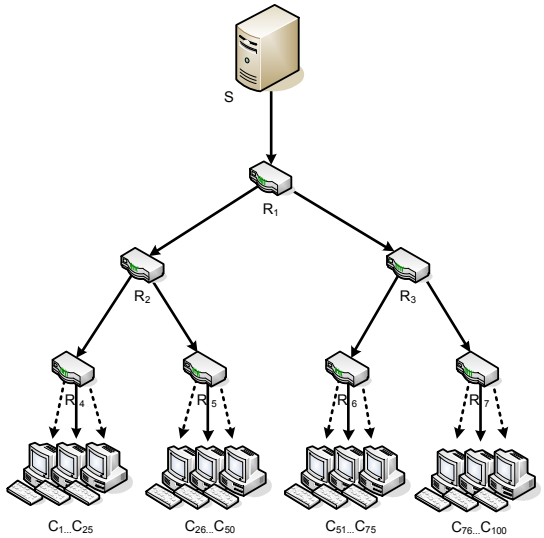
Messergebnisse Binärbaum



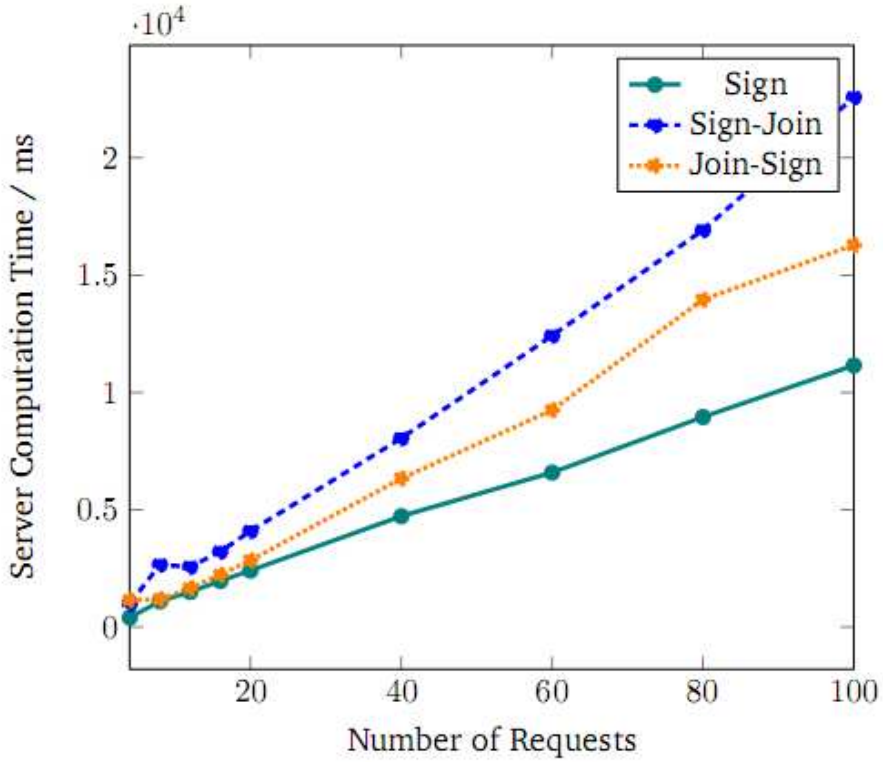
- Jeweils 25 Clients mit R_4…R_7 verbunden
- Jeweils ein Request pro Client
- Requests bzw. Responses werden gemäß „Round-Robin“ auf Clients von R_4…R_7 verteilt



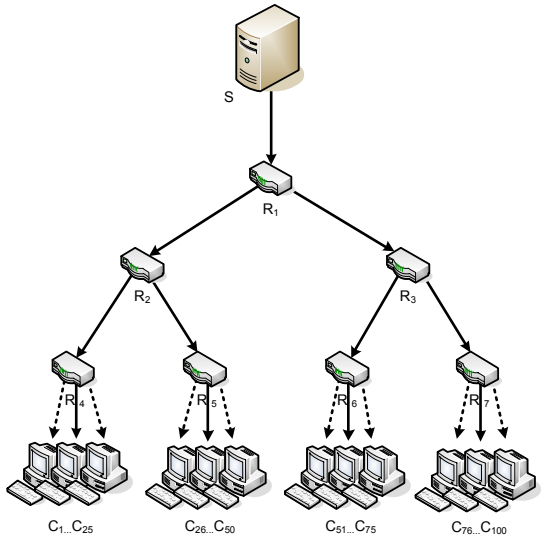
Messergebnisse Binärbaum



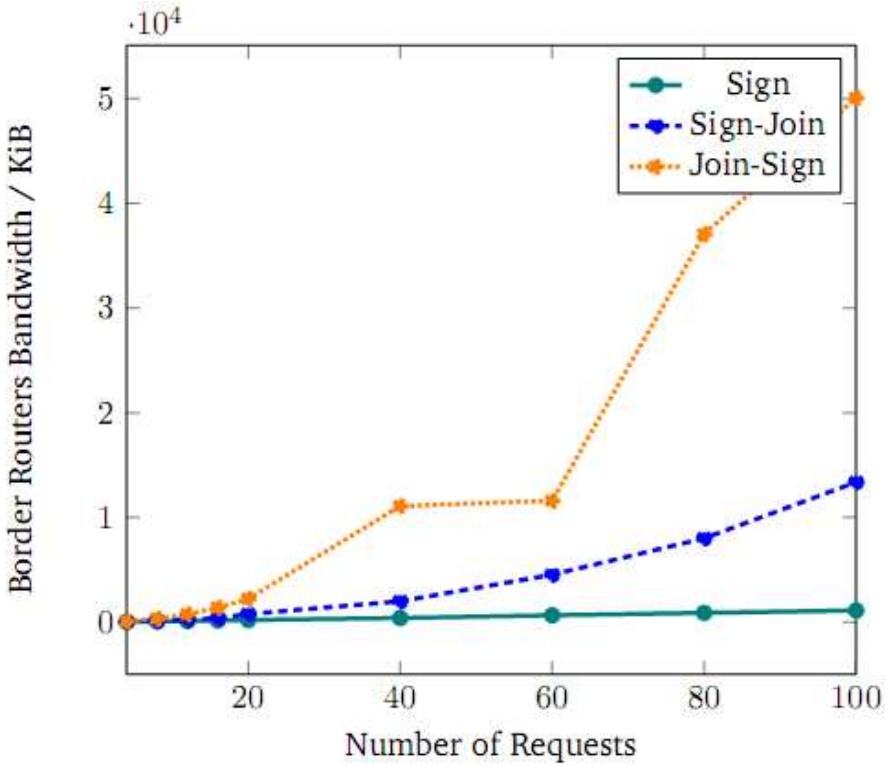
- Jeweils 25 Clients mit R_4…R_7 verbunden
- Jeweils ein Request pro Client
- Requests bzw. Responses werden gemäß „Round-Robin“ auf Clients von R_4…R_7 verteilt



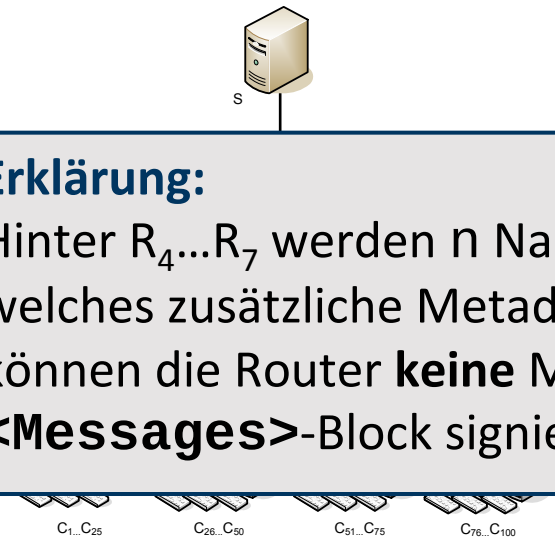
Messergebnisse Binärbaum



- Jeweils 25 Clients mit R_4…R_7 verbunden
- Jeweils ein Request pro Client
- Requests bzw. Responses werden gemäß „Round-Robin“ auf Clients von R_4…R_7 verteilt



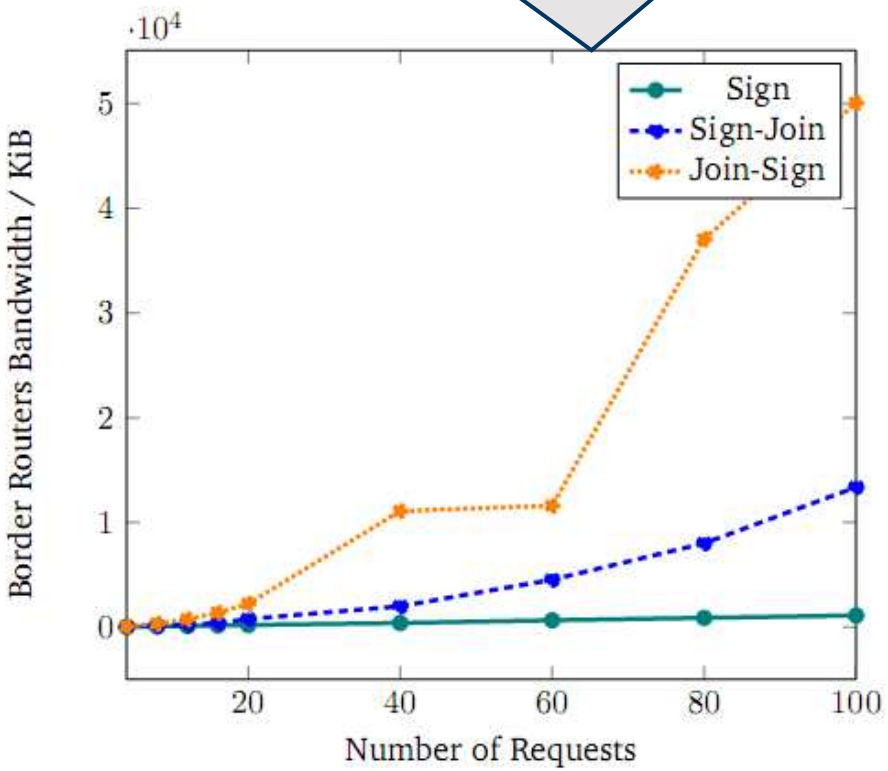
Messergebnisse

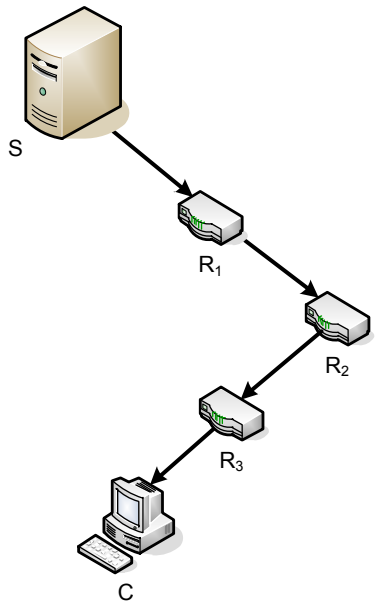


Erklärung:
Hinter R_4…R_7 werden n Nachrichten im AGX-Format verschickt, welches zusätzliche Metadaten enthält. Insbesonders bei Join-Sign können die Router **keine** Metadaten verwerfen, weil der gesamte **<Messages>**-Block signiert ist!

…nden

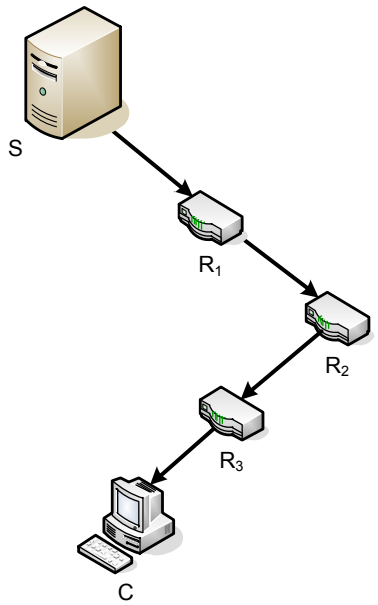
…gemäß „Round-Robin" …R_4…R_7 verteilt





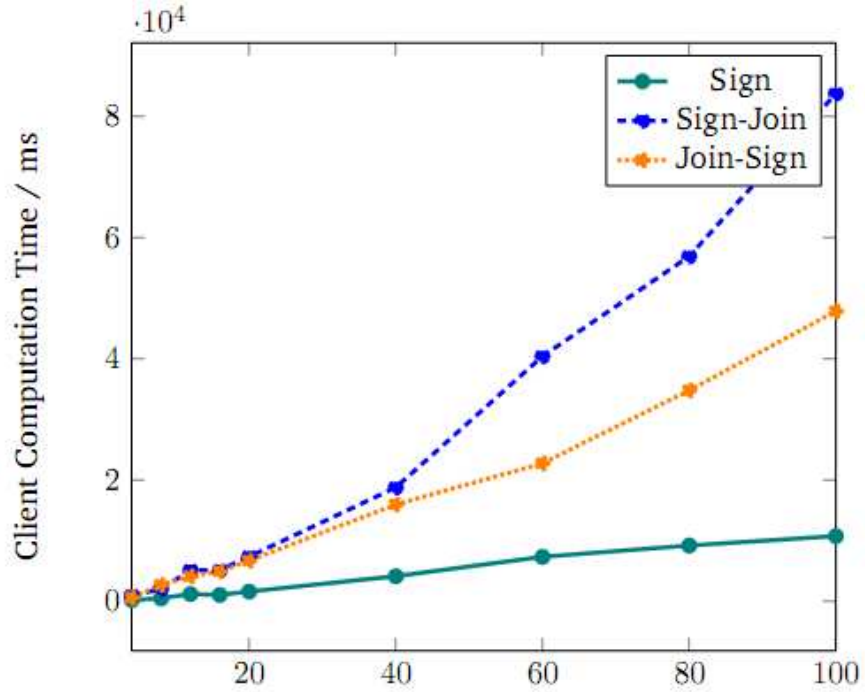
Messergebnisse Supply Chain

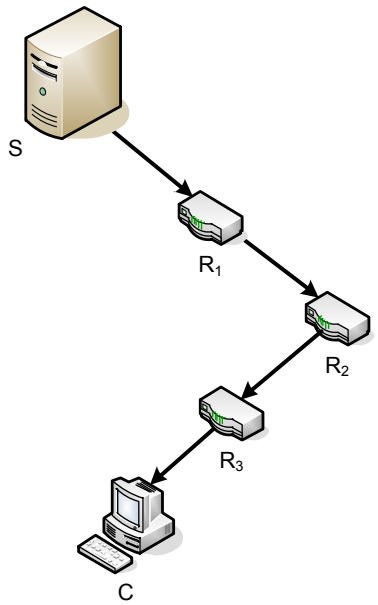
- Ein einziger Client stellt n Anfragen
- Client über eine Kette von Routern erreichbar
- Keine „explodierende“ Bandbreite bei den Routern mehr zu erwarten!



Messergebnisse Supply Chain

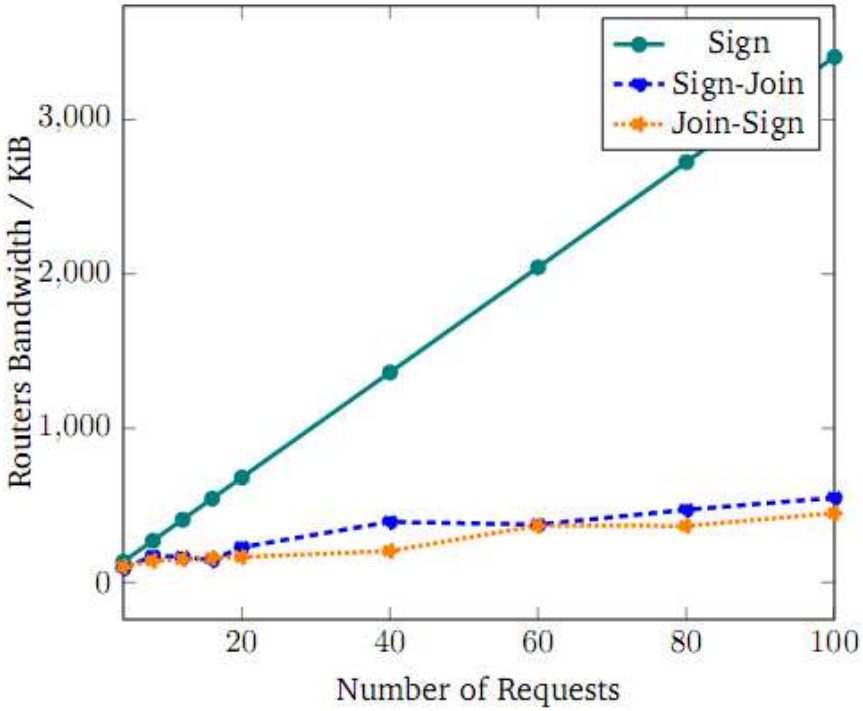
- Ein einziger Client stellt n Anfragen
- Client über eine Kette von Routern erreichbar
- Keine „explodierende“ Bandbreite bei den Routern mehr zu erwarten!





Messergebnisse Supply Chain

- Ein einziger Client stellt n Anfragen
- Client über eine Kette von Routern erreichbar
- Keine „explodierende“ Bandbreite bei den Routern mehr zu erwarten!



Abschluss

Zusammenfassung

Zusammenfassung

- Einführung in XML Signature, Ähnlichkeit und Aggregation von XML-Dokumenten

Zusammenfassung

- Einführung in XML Signature, Ähnlichkeit und Aggregation von XML-Dokumenten
- Nachrichtenformat AGX und dessen Verarbeitung

Zusammenfassung

- Einführung in XML Signature, Ähnlichkeit und Aggregation von XML-Dokumenten
- Nachrichtenformat AGX und dessen Verarbeitung
- Zwei Ansätze zur Kombination von Aggregation und Signatur: **`Sign-Join`** und **`Join-Sign`**
  - Verarbeitung und Sicherheitsanalyse

Zusammenfassung

- Einführung in XML Signature, Ähnlichkeit und Aggregation von XML-Dokumenten
- Nachrichtenformat AGX und dessen Verarbeitung
- Zwei Ansätze zur Kombination von Aggregation und Signatur: **`Sign-Join`** und **`Join-Sign`**
  - Verarbeitung und Sicherheitsanalyse
- AGXcast-Protokoll

Zusammenfassung

- Einführung in XML Signature, Ähnlichkeit und Aggregation von XML-Dokumenten
- Nachrichtenformat AGX und dessen Verarbeitung
- Zwei Ansätze zur Kombination von Aggregation und Signatur: **`Sign-Join`** und **`Join-Sign`**
  - Verarbeitung und Sicherheitsanalyse
- AGXcast-Protokoll
- Framework für Remote Performance-Messungen

Zusammenfassung

- Einführung in XML Signature, Ähnlichkeit und Aggregation von XML-Dokumenten
- Nachrichtenformat AGX und dessen Verarbeitung
- Zwei Ansätze zur Kombination von Aggregation und Signatur: **`Sign-Join`** und **`Join-Sign`**
  - Verarbeitung und Sicherheitsanalyse
- AGXcast-Protokoll
- Framework für Remote Performance-Messungen

  Parameterwahl und Messergebnisse

Fazit

Fazit

- Erhebliche Einsparung von BB durch Aggregation

Fazit

- Erhebliche Einsparung von BB durch Aggregation
  - Dies kauft man mit zusätzlicher Rechenlast auf Sender- (ggf. Vermittler-) und Empfängerseite ein

Fazit

- Erhebliche Einsparung von BB durch Aggregation
  - Dies kauft man mit zusätzlicher Rechenlast auf Sender- (ggf. Vermittler-) und Empfängerseite ein
- **`Join-Sign`** *besser* als **`Sign-Join`** bzgl. Rechenzeit und Bandbreitenbedarf
  - n XPath-Ausdrücke, aber nur eine Signatur pro Aggregatnachricht!

Fazit

- Erhebliche Einsparung von BB durch Aggregation
  - Dies kauft man mit zusätzlicher Rechenlast auf Sender- (ggf. Vermittler-) und Empfängerseite ein
- **`Join-Sign`** *besser* als **`Sign-Join`** bzgl. Rechenzeit und Bandbreitenbedarf
  - n XPath-Ausdrücke, aber nur eine Signatur pro Aggregatnachricht!
- Problem: Verzweigende Netzw.topologien, BB der *last hops* sehr hoch!
  - Umstellen auf Unicast-Nachrichten
  - Problem mit **`Join-Sign`**: Router muss Signatur verifizieren!

    `Sign-Join` auch bei Aggregation nur auf Teilstrecke anwendbar

RUB

Vielen Dank nicht nur für die Aufmerksamkeit!

Fragen oder Anmerkungen ?

Quellen

1. Azzini et al., in *SWS*, 2009, **Extending the Similarity-Based XML Multicast Approach with Digital Signatures**
2. Phan et al., in *IEEE T. Services Computing* 1.2, 2008, **Similarity-based SOAP Multicast Protocol to Reduce Bandwidth and Latency in Web Services**
3. Bartel et al., 2008, **XML Signature Syntax and Processing (Second Edition)**, online [accessed on 16-09-2010], url: http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
4. Boivie et al., 2007, **RFC 5058 - Explicit Multicast (Xcast) Concepts and Options**, online [accessed on 16-09-2010], url: http://www.ietf.org/rfc/rfc5058.txt