Modeling Security Aspects of Network Aggregation Protocols

Frank Werner **Raoul Steffen**

Fachgespräch Sensornetze 2009
14. August 2009

ZeuS

Motivation

- Formale Methoden: *„… Einsatz mathematischer Modelle und Techniken zur Analyse von Informations- und Kommunikationstechnologien“*

- Sensornetze:
  - Möglichkeiten eines Angreifers Betrieb zu manipulieren (non-det.)
  - drahtlose Kommunikation (non-det.)
  - Ressourcenbeschränkung

- Formale Methoden (MC Verfahren) + Sensornetze
  → sichere und korrekte Protokolle (formal bewiesen)
  → Steigerung der Software-Qualität

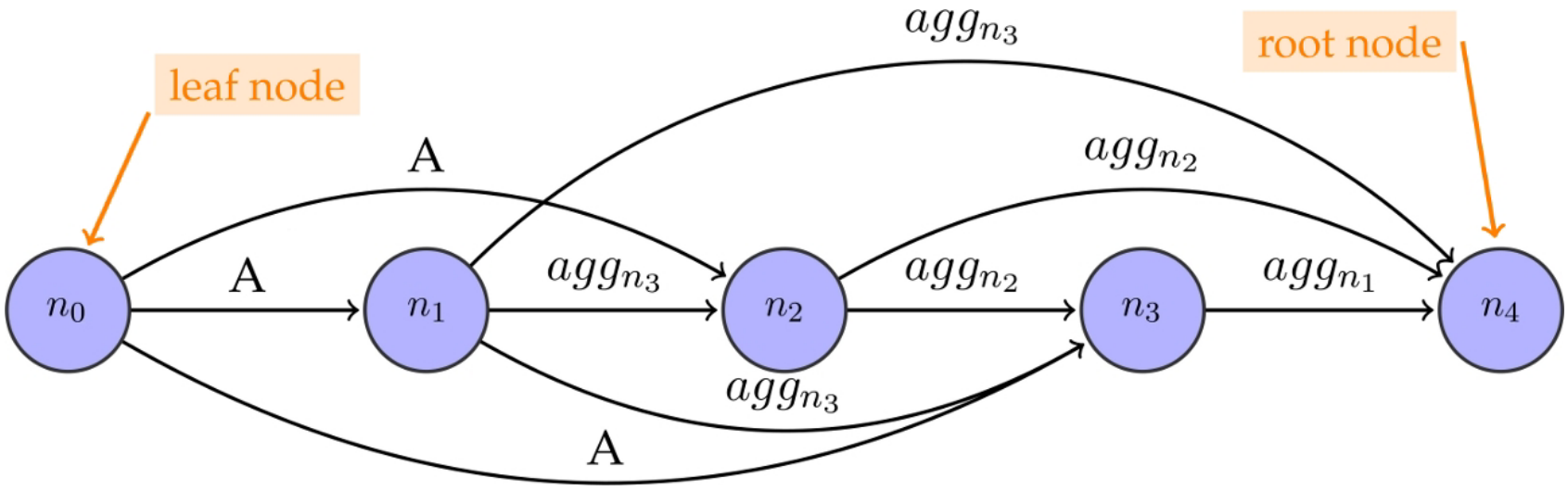
Übersicht

- Concast Protokoll (ESAWN)
- Spin Tool
- Modellierung
  - Angreifer
  - Protokoll in Promela
  - spezifizierte Eigenschaften
- Ergebnisse

ESAWN[1]

- Daten werden aggregiert (sicherer probabilistischer Concast)
- Knoten überprüfen gegenseitig ob korrekt aggregiert
- Beispiel: Knoten n_2
  - überprüft Aggregate von n_0 und n_1
    - Aggregat gefälscht → Alarm
    - Aggregat korrekt → berechne neues Aggregat agg_{n2}
  - wird durch Knoten (Zeugen) n_3 und n_4 überprüft
- Energieverbrauch verringern: überprüfe nur mit p%

[1] Erik-Oliver Blaß, Joachim Wilke, Martina Zitterbart. *A Security-Energy Trade-Off for Authentic Aggregation in Sensor Networks*, SECON 2009, USA



SPIN

- Spin Model Checker (explicit state)
  - Modellierung paralleler Prozesse und verteilten Anwendungen
  - asynchrone/synchrone Kanäle
  - lokale oder globale Variablen/Datenstrukturen
  - atomare Blöcke
- Eigenschaften in LTL spezifiziert

  $\Phi ::= ap \mid \neg\Phi \mid \Phi v \Psi \mid X\Phi \mid \Phi U \Psi \mid \Box\Phi \mid \Diamond\Phi$
- Verifikation: Modell + LTL -> Büchi-Automat

Verwandte Arbeiten

- Slede Tool[2]: Modelle in TinyOS 1.0 als Eingabe→ Generiert Promela Modell

[2] Youssef Hanna. Slede: *lightweight verification of sensor network security protocol implementations*. In ESEC-FSE companion '07: The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, New York, USA, 2007

- Klee Net[3]: generiert Tests mit hoher Abdeckung

[3] Cristian Cadar and Daniel Dunbar and Dawson R. Engler : KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs, 8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008, December 8-10, USA

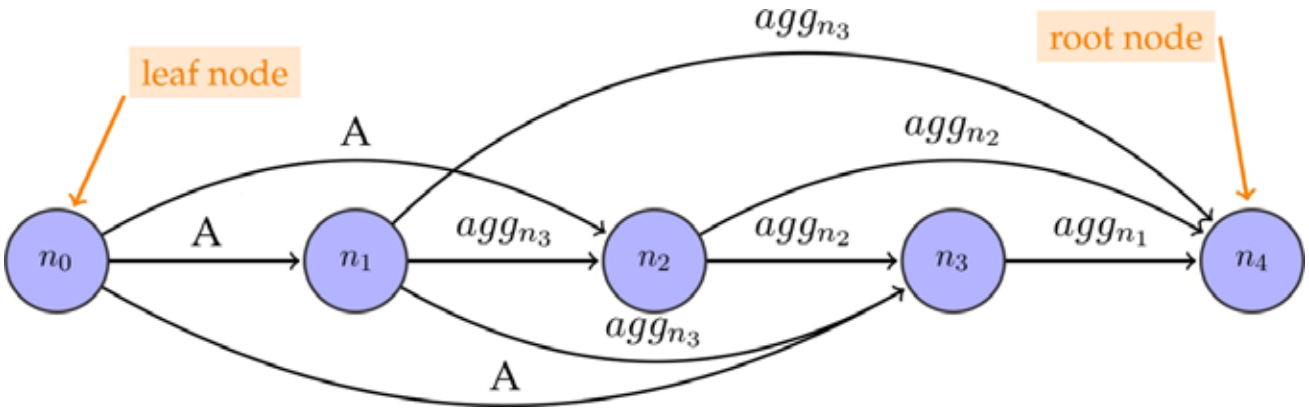
Definition: Angreifer

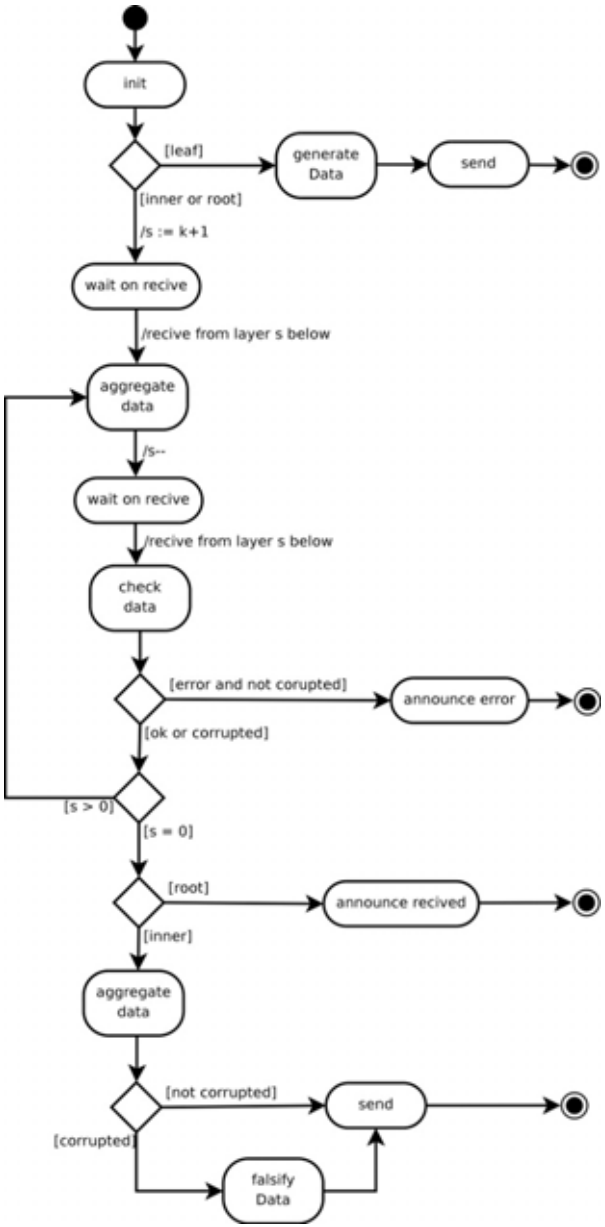
- Kann Knoten unbemerkt kompromittieren
- kompromittierte Knoten
  - können Aggregate fälschen
  - kooperieren
  - bleiben unbemerkt, bis sie aktiv werden

SPIN Modell

- Blatt Knoten und Root Knoten fix
- übrige Knoten werden nicht-deterministisch korrupt oder nicht-korrupt
- Blatt Knoten initialisiert Lauf
- Empfang von Paketen:
  - Knoten vergleicht Ist mit Soll

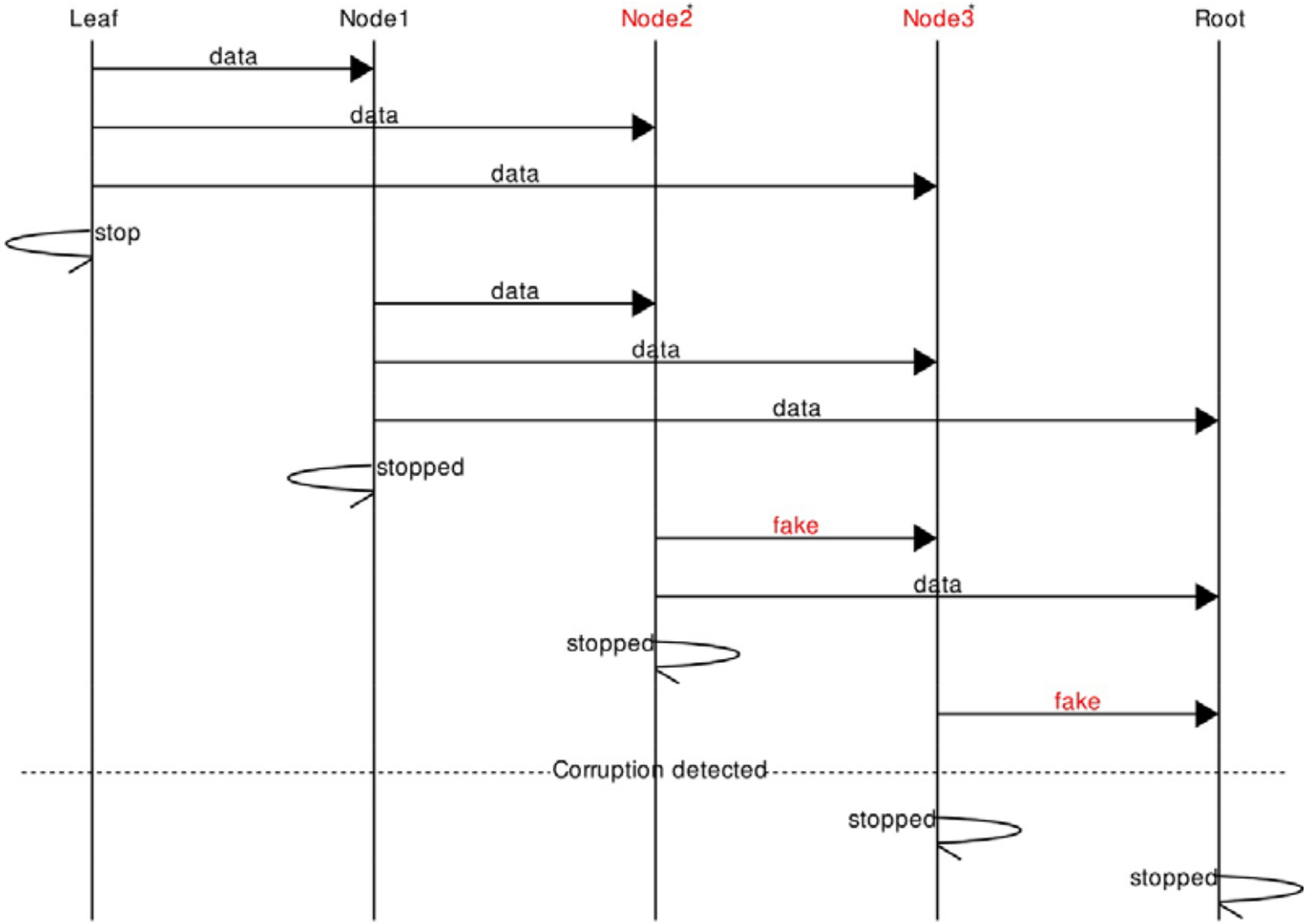
  →Aggregat wird weitergeleitet/Alarm





Ablauf des Protokolls

Scenario: Korrekter Lauf wobei Täuschung erkannt wird



Eigenschaften

- Irgendwann empfängt Root Knoten Daten (korrektes/gefälschtes Paket oder Alarm Nachricht):

  ◇(AnnounceRcv v CheatDetect)

- Immer wenn der Root Knoten ein Paket empfangen hat, so ist es korrekt oder die Fälschung wird erkannt:

  ☐ AnnounceRcv → (RcvDataCorrect v CheatDetect)

- Immer wenn ein Paket gefälscht wird, so wird dies irgendwann einmal berichtet:

  ☐ (FakePacketsSnd > 0 → ◇CheatDetect)

- Immer wenn Daten korrekt empfangen wurden, so hat keiner der Knoten betrogen:

  ☐ (RcvDataCorrect →!(FakeNodes>0))

Ergebnisse

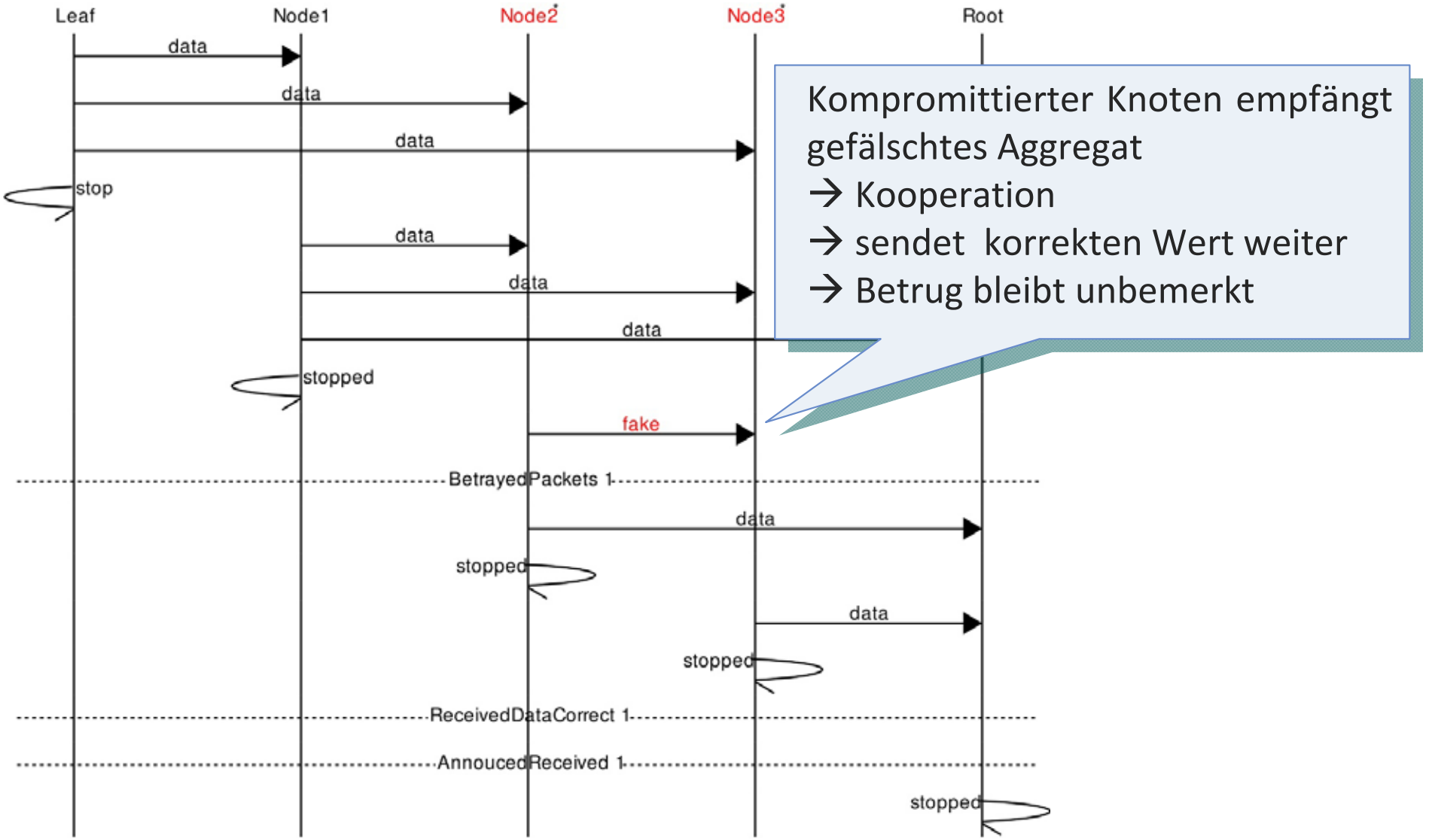
Eigenschaften über Root Knoten

Eigenschaft	Ergebnis
◇(AnnounceRcv v CheatDetect)	gültig
☐ AnnounceRcv → (RcvDataCorrect v CheatDetect)	gültig
☐(FakePacketsSnd>0 →◇CheatDetect)	ungültig
☐ (RcvDataCorrect →!(FakeNodes>0))	ungültig

globale Eigenschaften

Ergebnisse

Eigenschaft 3 und 4 ungültig



Schlussfolgerung

- Keine Fehler in Protokoll gefunden
- Ansatz der Modellprüfung gut geeignet um simultanes Verhalten von Protokollen zu analysieren
- manuelle Modellierung aufwendig und fehleranfällig

Neuer Ansatz:

→automatische Modell Generierung aus TinyOS

→Abstraktion

→Software Verifikation mittels Software Bounded Model Checking [5]

[5] Frank Werner, Hendrik Post: *Embedded Software Correctness by Bounded Model Checking*, publication in progress

Vielen Dank für
Ihre Aufmerksamkeit.