Detecting Misbehavior in Wireless Sensor Networks

Nils Knappmeier

IT Security Group Departement of Computer Science
University of Technology Darmstadt

20.1.2005

Contents

Contents

Common challenges

- Energy constraints
  - Sensor running on battery
  - Not likely to get a new battery soon
- Resource constraints
  - Little main memory
  - Small processing unit
- Autonomy
  - User is not nearby

What is misbehavior detection. . .

. . . and why is it important?

Even with

- encrypted communication and
- authentificated communication

Attacker may have physical access to sensor nodes!

- Extraction of cryptographic keys
- Wormhole, Blackhole,. . . attacks possible again

What is misbehavior detection. . .
. . . and why is it important?

Even with

- encrypted communication and
- authentificated communication

Attacker may have physical access to sensor nodes!

- Extraction of cryptographic keys
- Wormhole, Blackhole,. . . attacks possible again

What is misbehavior detection. . .

. . . and why is it important?

Even with

- encrypted communication and
- authentificated communication

Attacker may have physical access to sensor nodes!

- Extraction of cryptographic keys
- Wormhole, Blackhole,. . . attacks possible again

What is misbehavior detection. . .

. . . and why is it important?

- Detect misbehaving nodes/compromised keys
  $\Rightarrow$ "Decentralized intrusion detection system for WSN"

- Handle intrusion when detected
  $\Rightarrow$ "Statistical enroute-filtering of injected false data"

What is misbehavior detection. . .

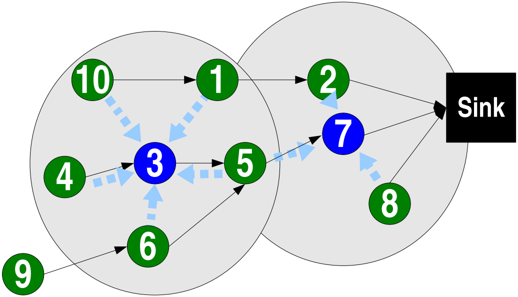
. . . and why is it important?

- Detect misbehaving nodes/compromised keys
  $\Rightarrow$ "Decentralized intrusion detection system for WSN"
- Handle intrusion when detected
  $\Rightarrow$ "Statistical enroute-filtering of injected false data"

Contents

Global Architecture

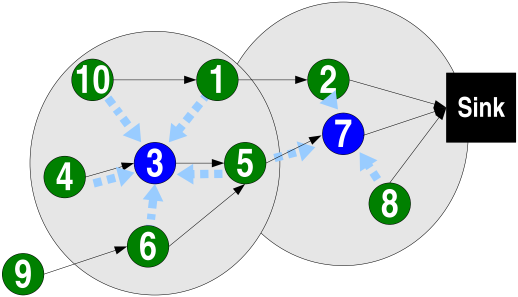
- Monitor nodes (3 , 7) use promiscous listening



- Nodes do not move
- Nodes can be identified
- Reliable connection from monitor to sink

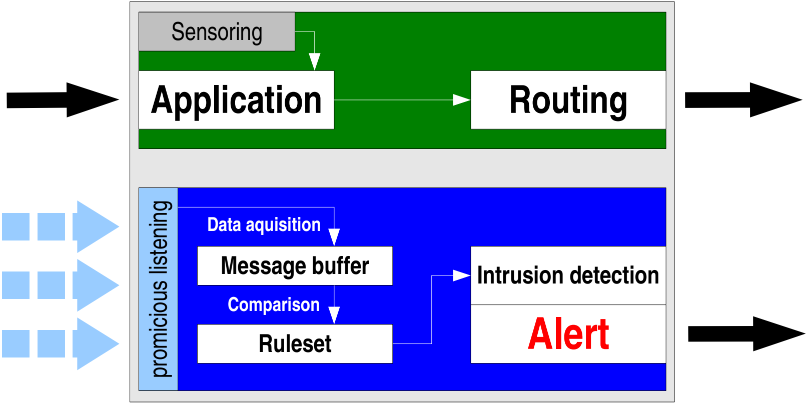
Global Architecture

- Monitor nodes (3, 7) use promiscous listening



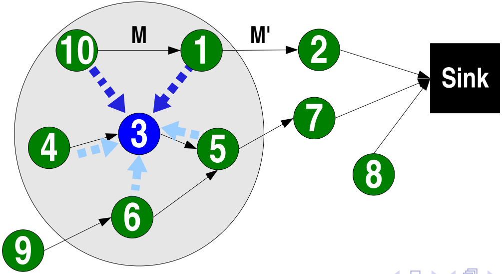
- Nodes do not move
- Nodes can be identified
- Reliable connection from monitor to sink

Monitor node



Retransmission, delay and integrity rule

Retransmission rule	Does 1 forward the message? Blackhole attack or selective forwarding
Integrity rule	$M = M'$? Message alteration attack
Delay rule	$t(M') - t(M) <$ treshold ? Message delay attack



Simulation setup

Size	Sensors	100 nodes
	Monitors	28 nodes
Procedure	Total duration	10000 iterations
	Learning phase	1000 iterations
	10 attack cycles with each	
	Idle time	700 iterations
	Attack duration	200 iterations
Simulated	One compromised node	
	One form of attack	
	Network failure rate	10% (20%)

Detection effectivness

Simulation results

Attack	Small Message Buffer DR	Small Message Buffer FP	Large Message Buffer DR	Large Message Buffer FP
Message delay	bad	few	good	hardly any
Blackhole	good	**too many**	good	few
Selective forwarding	medium	**too many**	good	few
Wormhole	good	many	good	few
Message repetition	good	few	good	hardly any
Jamming	good	medium	good	few
Data alteration	good	**too many**	medium	few

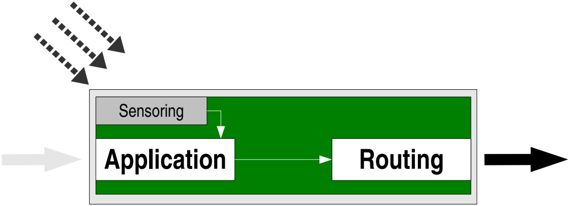
DR=Detection Rate **FP**=False Positives

Contents

Motivation...

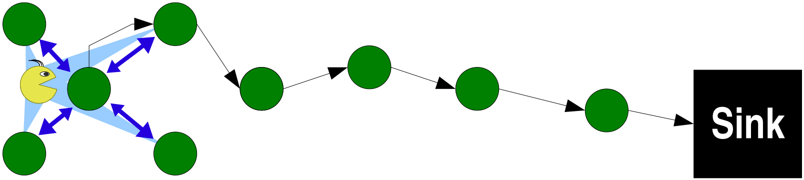
...of the statistical enroute-filtering

What if **initial report** is already fabricated?



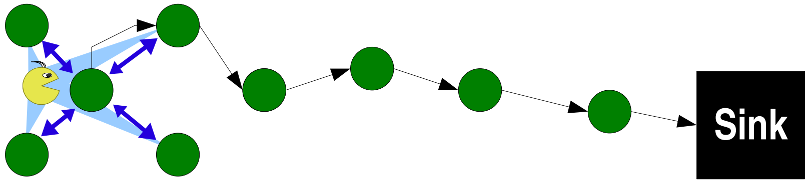
Goal:
Recognition and early disposal of fabricated reports

General approach



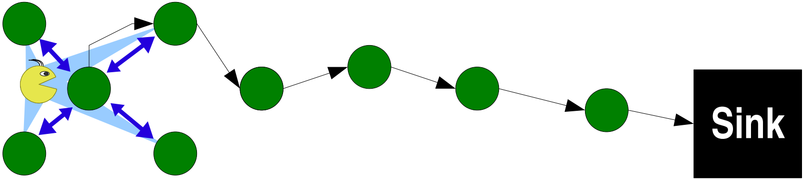
- Verification: **En-route** (to save energy) and **at the sink**
- How to distribute the keys?

General approach



- Verification: **En-route** (to save energy) and **at the sink**
- How to distribute the keys?

General approach

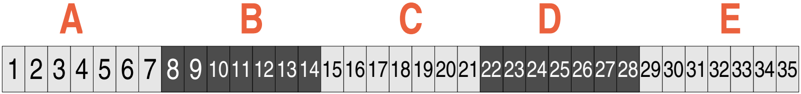


- Verification: **En-route** (to save energy) and **at the sink**
- How to distribute the keys?

Keys, categories, index numbers

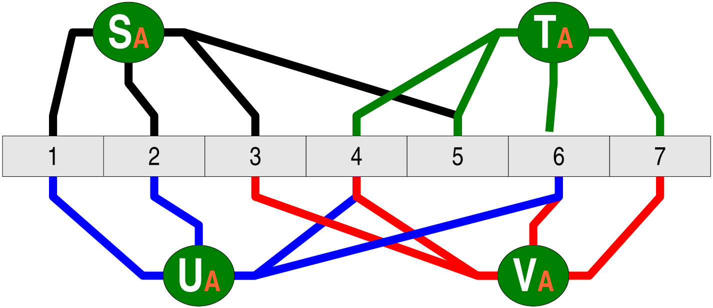
- Global key pool
- Numbering, Partitioning

Keys, categories, index numbers



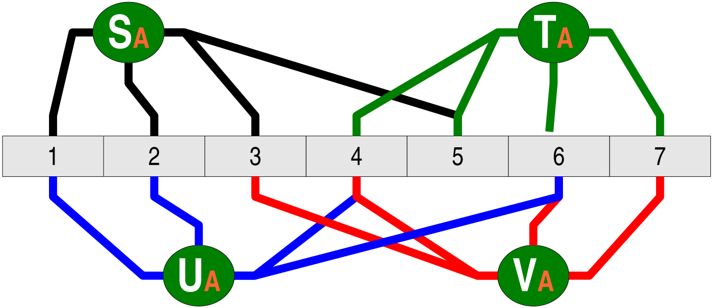
- Global key pool
- Numbering, Partitioning

A node stores 4 random keys from the same category



- Node S stores: $\{(1, K_1), (2, K_2), (3, K_3), (5, K_5)\}$
- Node T stores: $\{(4, K_4), (5, K_5), (6, K_6), (7, K_7)\}$
- Node U stores: $\{(1, K_1), (2, K_2), (4, K_4), (6, K_6)\}$
- Node V stores: $\{(3, K_3), (4, K_4), (6, K_6), (7, K_7)\}$

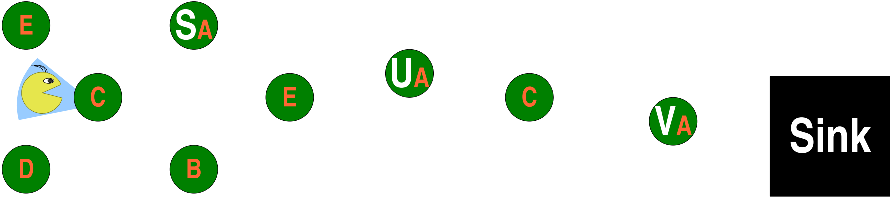
A node stores 4 random keys from the same category



- Node **S** stores: $\{(1, K_1), (2, K_2), (3, K_3), (5, K_5)\}$
- Node **T** stores: $\{(4, K_4), (5, K_5), (6, K_6), (7, K_7)\}$
- Node **U** stores: $\{(1, K_1), (2, K_2), (4, K_4), (6, K_6)\}$
- Node **V** stores: $\{(3, K_3), (4, K_4), (6, K_6), (7, K_7)\}$

Report generation

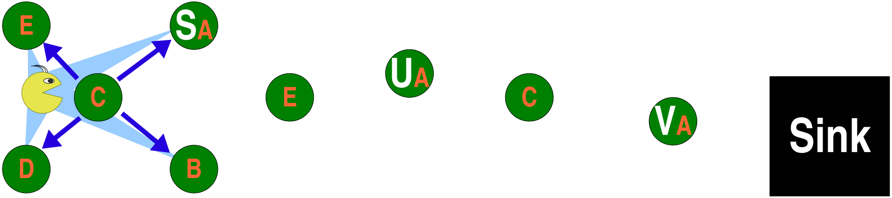
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- C detects stimulus
- $report = (pos, timestamp, type)$ is verified
- Neighors return $(i, MAC(report, K_i))$

Report generation

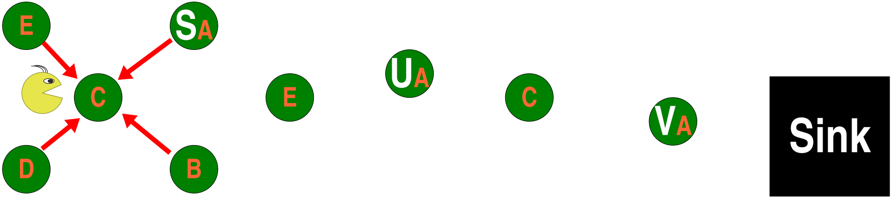
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- C detects stimulus
- *report* $= (pos, timestamp, type)$ is verified
- Neighors return $(i, MAC(report, K_i))$

Report generation

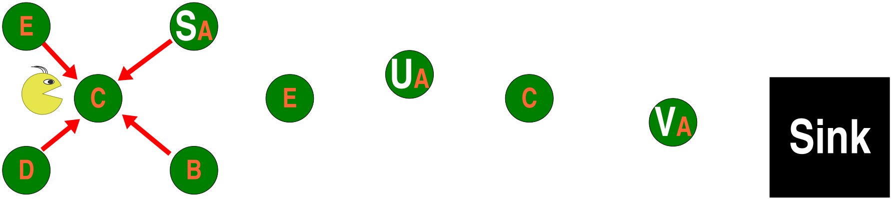
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- C detects stimulus
- $report = (pos, timestamp, type)$ is verified
- Neighors return $(i, MAC(report, K_i))$

Report generation

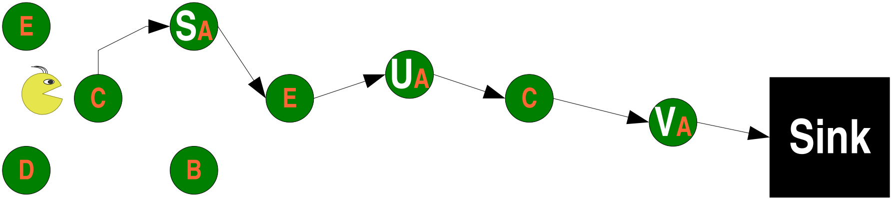
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- C selects 3 MACs each of a different category
- C sends report to sink, with MACs attached

Report generation

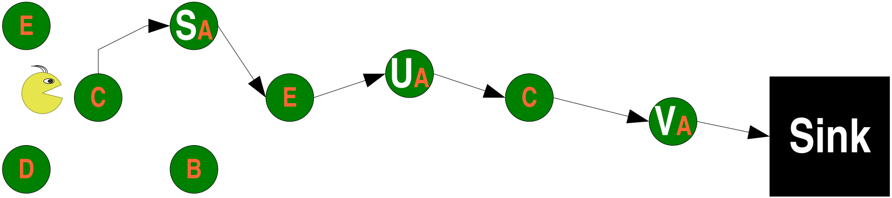
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- C selects 3 MACs each of a different category
- C sends report to sink, with MACs attached

Statistical en-route filtering

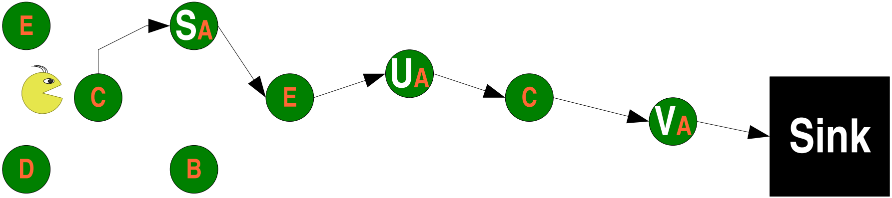
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- 2 *MAC*s from the same category? ⇒ Drop
- Invalid *MAC* found? ⇒ Drop
- *MAC*s not verifiable or correct? ⇒ Forward

Statistical en-route filtering

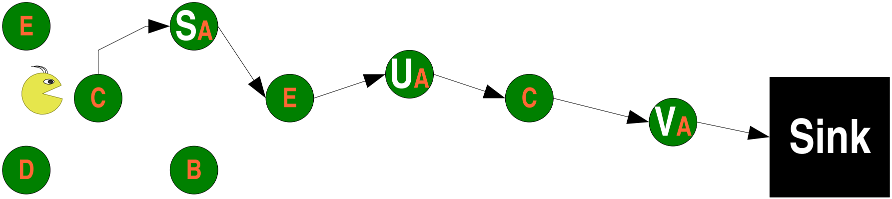
$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$



- 2 *MAC*s from the same category? ⇒ Drop
- Invalid *MAC* found? ⇒ Drop
- *MAC*s not verifiable or correct? ⇒ Forward

Filtering at the sink

$$(pos, timestamp, type), (2, MAC_2), (10, MAC_{10}), (17, MAC_{17})$$



- Sink knows all keys of every category
- Verification **all** MACs attached to the report

Verification **all** MACs attached to the report

Theoretical efficiency estimate

Keys	Total number	1000 keys
	10 categories	100 keys
	Each node	50 keys
	Each report	5 MACs

Assuming the attacker has compromised $N_c < 5$ nodes.

- How likely that a node can identify a forged key?
- How likely that a forged key is identified after h hops?

Theoretical efficiency estimate

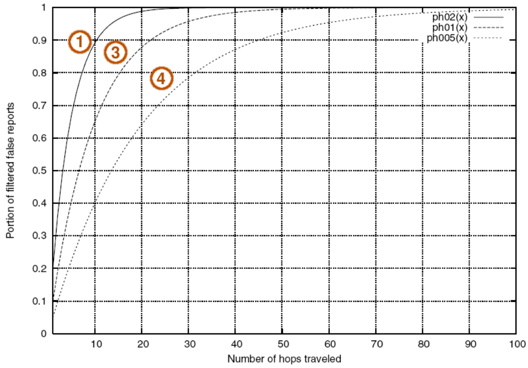
Keys	Total number	1000 keys
	10 categories	100 keys
	Each node	50 keys
	Each report	5 MACs

Assuming the attacker has compromised $N_c < 5$ nodes.

- How likely that a node can identify a forged key?
- How likely that a forged key is identified after h hops?

Packets dropped after *n* hops. . .

. . . for 1,3 and 4 compromised categories



Contents

Conclusion

1. Misbehavior detection in sensor networks is possible
   - Intrusion detection works for most attacks
   - False injection detection also works
2. Both systems have open issues
   - Intrusion detection and encrypted communication
   - Alerting the sink
   - En-route-filtering adresses only a single attack
3. Only systems for special aspects! Combination possible?
4. Evaluation mostly by simulation → level-of-detail

Conclusion

1. Misbehavior detection in sensor networks is possible
   - Intrusion detection works for most attacks
   - False injection detection also works
2. Both systems have open issues
   - Intrusion detection and encrypted communication
   - Alerting the sink
   - En-route-filtering adresses only a single attack
3. Only systems for special aspects! Combination possible?
4. Evaluation mostly by simulation $\rightarrow$ level-of-detail

Conclusion

1. Misbehavior detection in sensor networks is possible
    - Intrusion detection works for most attacks
    - False injection detection also works
2. Both systems have open issues
    - Intrusion detection and encrypted communication
    - Alerting the sink
    - En-route-filtering adresses only a single attack
3. Only systems for special aspects! Combination possible?
4. Evaluation mostly by simulation $\rightarrow$ level-of-detail

Conclusion

1. Misbehavior detection in sensor networks is possible
   - Intrusion detection works for most attacks
   - False injection detection also works
2. Both systems have open issues
   - Intrusion detection and encrypted communication
   - Alerting the sink
   - En-route-filtering adresses only a single attack
3. Only systems for special aspects! Combination possible?
4. Evaluation mostly by simulation $\rightarrow$ level-of-detail

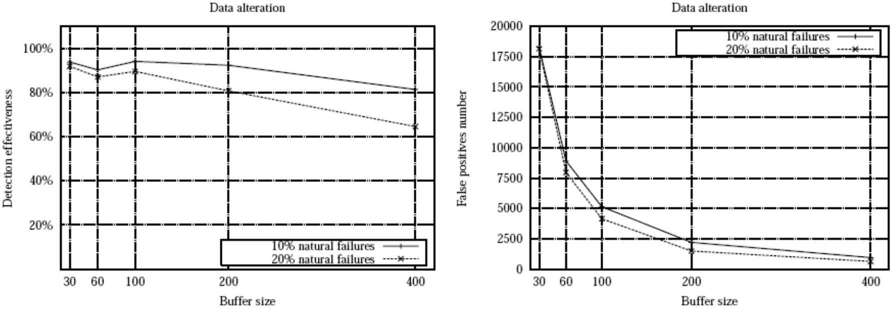
Conclusion

1. Misbehavior detection in sensor networks is possible
   - Intrusion detection works for most attacks
   - False injection detection also works
2. Both systems have open issues
   - Intrusion detection and encrypted communication
   - Alerting the sink
   - En-route-filtering adresses only a single attack
3. Only systems for special aspects! Combination possible?
4. Evaluation mostly by simulation → level-of-detail

Quellen

- Decentralized Intrusion Detection in Wireless Sensor Networks, Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno P. S. Rocha, Antonio A. F. Loureiro, Linnyer B. Ruiz, Hao Chi Wong, October 2005, Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks Q2SWinet '05
- Statistical En-route Filtering of Injected False Data in Sensor Networks, Fan Ye, Haiyun Luo, Songwu Lu, Lixia Zhang, UCLA Computer Science Departement, Los Angeles

Energy consumption



Energy consumption

- Base values for energy consumption
  - Listening: $0.01 \frac{mJ}{message}$
  - Receiving: $0.15 \frac{mJ}{message}$
  - Sending: $0.48 \frac{mJ}{message}$
- Results
  - Monitor nodes consume more energy
  - Nodes near the sink consume more energy

Energy consumption

- Base values for energy consumption
  - Listening: $0.01 \frac{mJ}{message}$
  - Receiving: $0.15 \frac{mJ}{message}$
  - Sending: $0.48 \frac{mJ}{message}$
- Results
  - Monitor nodes consume more energy
  - Nodes near the sink consume more energy