SR3: Secure Reputation-based Resilient Routing

Karine Altisen   Stéphane Devismes
Raphaël Jamet   Pascal Lafourcade

VERIMAG, Universités de Grenoble

This work was supported by the ARESA2 ANR Project

Outline

Outline

Challenges of secure routing in WSN

Security for routing in wireless sensor networks is hard, but necessary:

- Low memory, computing power, energy consumption
- Wireless medium is inherently vulnerable
- Compromise nodes: easy

Two main types of attacks

Packet-level attacks, *e.g.*

- Data messages alteration
- Creation of new data or control messages

Two main types of attacks

Packet-level attacks, *e.g.*

- Data messages alteration
- Creation of new data or control messages

They are solved with lightweight cryptography (*e.g.*, hash functions, symmetric cryptography, nonces).

Two main types of attacks

Packet-level attacks, *e.g.*

- Data messages alteration
- Creation of new data or control messages

They are solved with lightweight cryptography (*e.g.*, hash functions, symmetric cryptography, nonces).

Routing-level attacks, *e.g.*

- Compromised nodes drop packets (blackholes, select forwarding)
- Attackers attract traffic using out-of-band channels (wormholes)

Two main types of attacks

Packet-level attacks, *e.g.*

- Data messages alteration
- Creation of new data or control messages

They are solved with lightweight cryptography (*e.g.*, hash functions, symmetric cryptography, nonces).

Routing-level attacks, *e.g.*

- Compromised nodes drop packets (blackholes, select forwarding)
- Attackers attract traffic using out-of-band channels (wormholes)

Our protocol is resilient against this type of attacks

Resiliency "Capacity of a network to endure and overcome internal attacks" [EOKMV11]

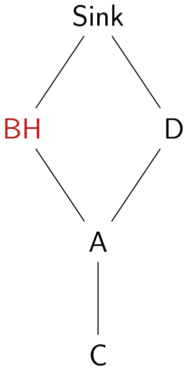
Outline

Main ideas

SR3: Secure Reputation-based Resilient Routing

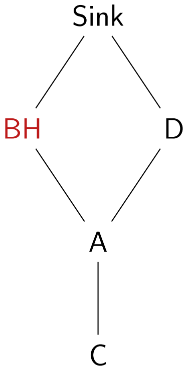
- Convergecast routing from all sensors to the sink (server)
- Reinforced random walk
  - Built with a reputation mechanism
  - Based on unconditionally trusted information

SR3: Overview



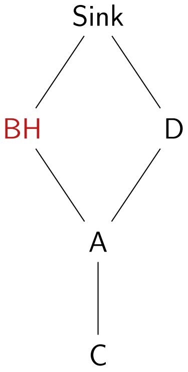
- A chooses the next hop among its neighbors, according to its confidence on them.

SR3: Overview



- A chooses the next hop among its neighbors, according to its confidence on them.
- The sink answers with an ACK that tries to follow the reverse of the path of the message.

SR3: Overview



- A chooses the next hop among its neighbors, according to its confidence on them.
- The sink answers with an ACK that tries to follow the reverse of the path of the message.
- If A gets a valid ACK, it increases its confidence in the neighbor who previously routed the corresponding message.

SR3: Packet-level attacks

Messages: $\mathcal{E}_{k_{src}}(Data||N), \mathcal{H}(N), Src$

ACK: N, Src

- Attacker who listens to the data
- Attacker who replays acknowledgements
- Attacker who alters or creates messages
- Attacker forging ACKs

SR3: Packet-level attacks

Messages: $\mathcal{E}_{k_{src}}(Data||N), \mathcal{H}(N), Src$

ACK: N, Src

- Attacker who listens to the data
  $\rightarrow$ Symmetric cryptography $\mathcal{E}_{k_{src}}$ using a key shared with the sink
- Attacker who replays acknowledgements
- Attacker who alters or creates messages
- Attacker forging ACKs

SR3: Packet-level attacks

Messages: $\mathcal{E}_{k_{src}}(Data||N), \mathcal{H}(N), Src$

ACK: N, Src

- Attacker who listens to the data
  $\rightarrow$ Symmetric cryptography $\mathcal{E}_{k_{src}}$ using a key shared with the sink
- Attacker who replays acknowledgements
  $\rightarrow$ An unpredictable nonce N per message, encrypted with the data
- Attacker who alters or creates messages
- Attacker forging ACKs

SR3: Packet-level attacks

Messages: $\mathcal{E}_{k_{src}}(Data||N), \mathcal{H}(N), Src$

ACK: N, Src

- Attacker who listens to the data
  $\rightarrow$ Symmetric cryptography $\mathcal{E}_{k_{src}}$ using a key shared with the sink
- Attacker who replays acknowledgements
  $\rightarrow$ An unpredictable nonce N per message, encrypted with the data
- Attacker who alters or creates messages
  $\rightarrow$ Add $\mathcal{H}(N)$ and check that it matches the ciphertext part
- Attacker forging ACKs

SR3: Packet-level attacks

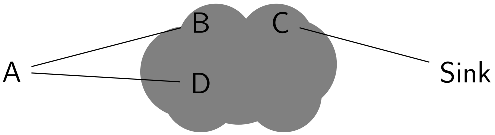
$$\text{Messages:} \quad \mathcal{E}_{k_{src}}(Data||N), \mathcal{H}(N), Src$$

$$\text{ACK:} \quad N, Src$$

- Attacker who listens to the data
  → Symmetric cryptography $\mathcal{E}_{k_{src}}$ using a key shared with the sink
- Attacker who replays acknowledgements
  → An unpredictable nonce N per message, encrypted with the data
- Attacker who alters or creates messages
  → Add $\mathcal{H}(N)$ and check that it matches the ciphertext part
- Attacker forging ACKs
  → Keep the nonce secret until delivery, and reveal it in the ACK

SR3: Keep track of messages using L_{Queue}

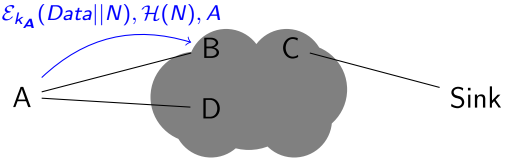
- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.



L_{Queue}: $[(N', D) \qquad]$

SR3: Keep track of messages using L_{Queue}

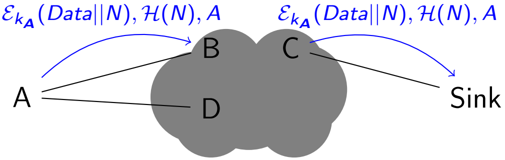
- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.



L_{Queue}: $[(N', D), (N, B)]$

SR3: Keep track of messages using L_{Queue}

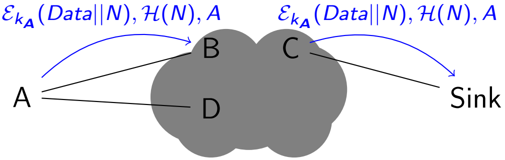
- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.



L_{Queue}: $[(N', D), (N, B)]$

SR3: Keep track of messages using L_{Queue}

- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.

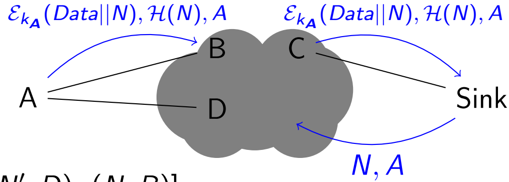


- Decrypt
- Check validity
- Build ACK

L_{Queue}: $[(N', D), (N, B)]$

SR3: Keep track of messages using L_{Queue}

- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.

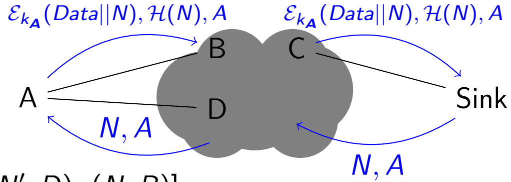


- Decrypt
- Check validity
- Build ACK

L_{Queue}: $[(N', D), (N, B)]$

SR3: Keep track of messages using L_{Queue}

- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.
- Upon reception of an ACK which contains N,
  - If A is not the final destination for that ACK, A routes it,
  - Else, if it recalls the corresponding message using L_{Queue}, it reinforces A's trust in B.
  - Otherwise, A drops the ACK.

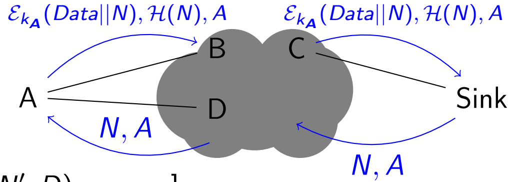


- Decrypt
- Check validity
- Build ACK

L_{Queue}: $[(N', D), (N, B)]$

SR3: Keep track of messages using L_{Queue}

- When A generates a new message with a nonce N, it chooses the next hop B, and stores a trace of this choice in L_{Queue}, a bounded size FIFO list.
- Upon reception of an ACK which contains N,
  - If A is not the final destination for that ACK, A routes it,
  - Else, if it recalls the corresponding message using L_{Queue}, it reinforces A's trust in B.
  - Otherwise, A drops the ACK.



- Decrypt
- Check validity
- Build ACK

L_{Queue}: $[(N', D)$ $]$

SR3: Reputation

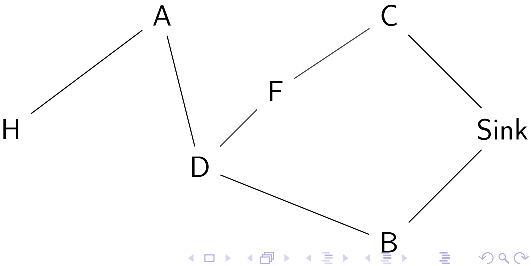
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[, ,] $(\star)$

$$P(X = C) = \frac{0 + 0.5}{1} = 50\%$$

$$P(X = D) = \frac{0 + 0.5}{1} = 50\%$$



SR3: Reputation

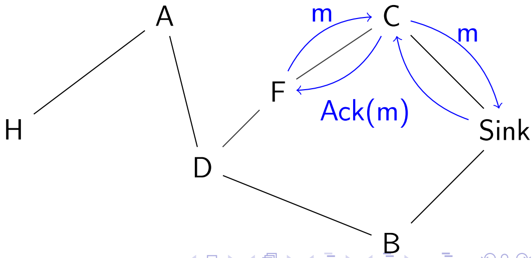
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[C, ,] $(\star)$

$$P(X = C) = \frac{1 + 0.5}{2} = 75\%$$

$$P(X = D) = \frac{0 + 0.5}{2} = 25\%$$



SR3: Reputation

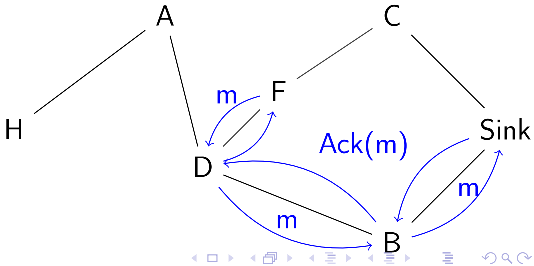
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[C, D,] (⋆)

$$P(X = C) = \frac{1 + 0.5}{3} = 50\%$$

$$P(X = D) = \frac{1 + 0.5}{3} = 50\%$$



SR3: Reputation

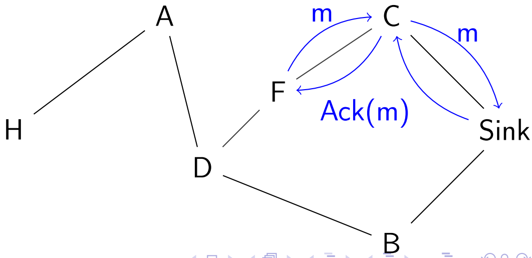
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[C, D, C] ($\star$)

$$P(X = C) = \frac{2 + 0.5}{4} = 63\%$$

$$P(X = D) = \frac{1 + 0.5}{4} = 37\%$$



SR3: Reputation

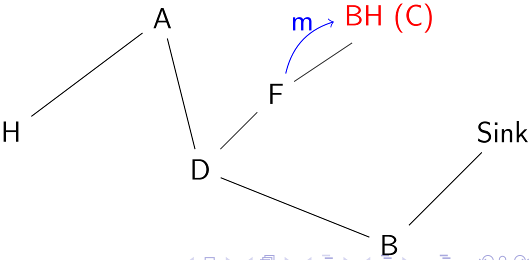
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[C, D, C] $(\star)$

$$P(X = C) = \frac{2 + 0.5}{4} = 63\%$$

$$P(X = D) = \frac{1 + 0.5}{4} = 37\%$$



SR3: Reputation

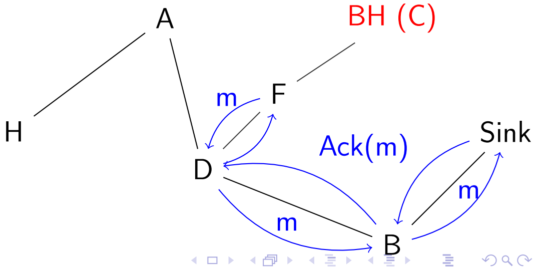
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[D, D, C] $(\star)$

$$P(X = C) = \frac{1 + 0.5}{4} = 37\%$$

$$P(X = D) = \frac{2 + 0.5}{4} = 63\%$$



SR3: Reputation

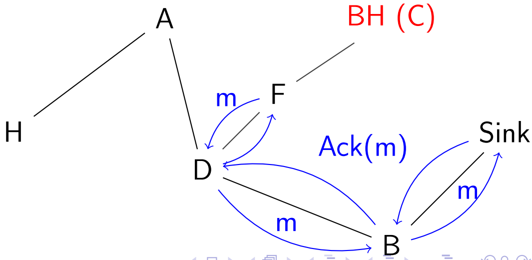
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[D, D, C] ($\star$)

$$P(X = C) = \frac{1 + 0.5}{4} = 37\%$$

$$P(X = D) = \frac{2 + 0.5}{4} = 63\%$$



SR3: Reputation

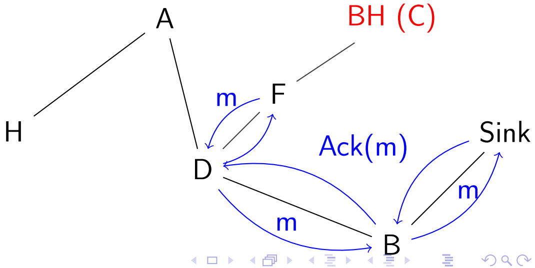
- Trust in a node is the number of identifiers of that node in a bounded FIFO list, $L_{Routing}$, initially empty.
- Messages are routed probabilistically according to the node's $L_{Routing}$.

$$Pr(X = n) = \frac{|L_{Routing}|_n + \delta_v^{-1}}{|L_{Routing}| + 1}$$

F's $L_{Routing}$, max size = 3 :
[D, D, D] $(\star)$

$$P(X = C) = \frac{0 + 0.5}{4} = 12\%$$

$$P(X = D) = \frac{3 + 0.5}{4} = 88\%$$

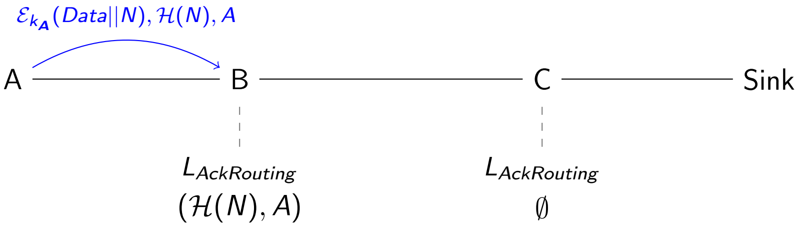


SR3: $L_{AckRouting}$ and routing acknowledgments

- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.

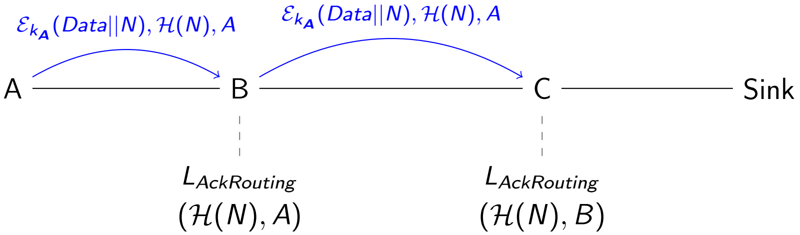
SR3: $L_{AckRouting}$ and routing acknowledgments

- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.



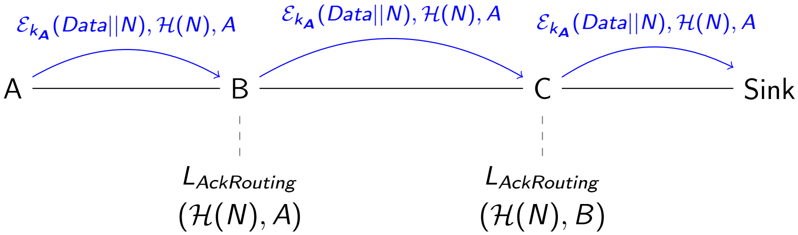
SR3: $L_{AckRouting}$ and routing acknowledgments

- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.



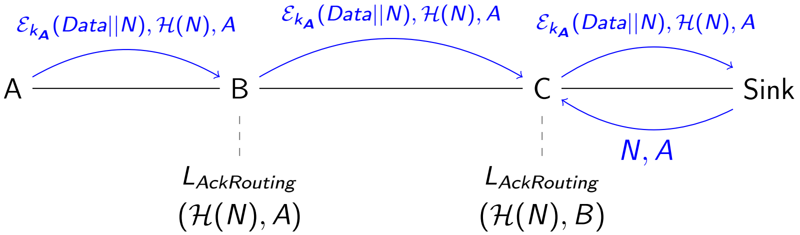
SR3: $L_{AckRouting}$ and routing acknowledgments

- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.



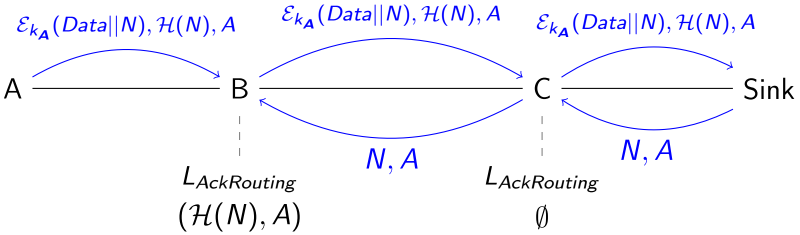
SR3: $L_{AckRouting}$ and routing acknowledgments

- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.
- ACKs follow those indications, and are routed randomly when they cannot.



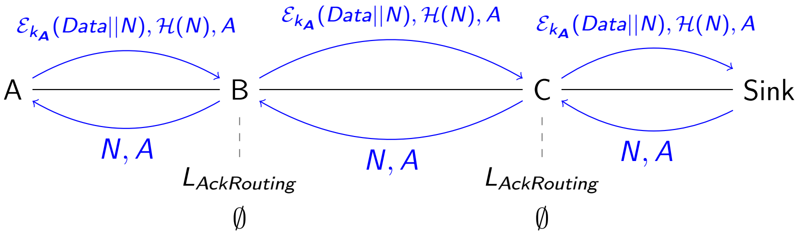
SR3: $L_{AckRouting}$ and routing acknowledgments

- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.
- ACKs follow those indications, and are routed randomly when they cannot.

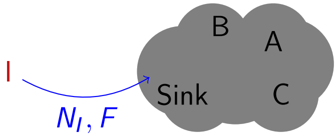


SR3: $L_{AckRouting}$ and routing acknowledgments

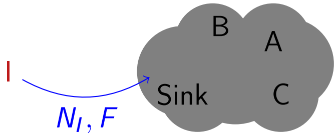
- Messages leave a trail of entries in a FIFO bounded-size list $L_{AckRouting}$ on the routing nodes.
- ACKs follow those indications, and are routed randomly when they cannot.



Lost and forged acknowledgements



Lost and forged acknowledgements



ACKs therefore have a $\frac{1}{Nb}$ probability of being dropped at each retransmission, with Nb an upper bound on the number of nodes.

Outline

Security properties and proofs

- Nonce confidentiality before delivery
- Data confidentiality, even after delivery
- Packet unforgeability (implies authentication and integrity)

Security properties and proofs

- Nonce confidentiality before delivery
- Data confidentiality, even after delivery
- Packet unforgeability (implies authentication and integrity)

These proofs are done using **games**, which evaluate the probability that an attacker breaks the property (its **advantage**), depending on the *primitives* used and the attacker *capabilities*.

Security properties and proofs

- Nonce confidentiality before delivery
- Data confidentiality, even after delivery
- Packet unforgeability (implies authentication and integrity)

These proofs are done using **games**, which evaluate the probability that an attacker breaks the property (its **advantage**), depending on the *primitives* used and the attacker *capabilities*.

Using game reductions, we are then able to find an upper bound on the advantage of any attacker, which we can then use to find the right compromise between costs and the level of security.

Modeling the protocol

Hash function

We model it as a random oracle $\mathcal{H} : \{0,1\}^{\eta_n} \rightarrow \{0,1\}^{\eta_h}$. The number of times the adversary calls $\mathcal{H}$ is denoted q_H.

Nonces

Nonces are modeled as truly random numbers of size η_n.

Block Cipher

A block cipher $\mathcal{F}$ is a set of permutations of $\{0,1\}^{\eta_c}$, indexed on keys. We assume that the block cipher is *PRP-CCA*-secure, which is the property we rely on for security.

$Gen_{src}^{\mathcal{O}(\cdot)}(Data)$

The function which generates a packet produced by *src* containing *Data*, using $\mathcal{O}$ as a block cipher, and drawing a new nonce N.

Data confidentiality

Let $\mathcal{A}$ be an adversary running in two phases ($\mathcal{A}_1$ and $\mathcal{A}_2$).

Experiment $\mathbf{Expt}_F^{FG}(\mathcal{A})$:

$k_{src} \xleftarrow{\$} Init(\mathcal{K})$

$(\mathcal{O}, \mathcal{O}^{-1}) \leftarrow (F_{k_{src}}, F_{k_{src}}^{-1})$

$(Data_0, Data_1, state) \xleftarrow{\$} \mathcal{A}_1^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}()$

$b \xleftarrow{\$} \{0, 1\}$

$(\langle C, h, src \rangle, N) \xleftarrow{\$} Gen_{src}^{\mathcal{O}(\cdot)}(Data_b)$

If $(b = \mathcal{A}_2^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}(\langle C, h, src \rangle, N, state))$

  Return 1

Else

  Return 0

Advantage and bounds

The find-then-guess advantage of $\mathcal{A}$ against F is defined as the probability of $\mathcal{A}$ winning the game (*i.e.* $Pr[\mathbf{Expt}_F^{FG}(\mathcal{A}) = 1]$) minus the probability of winning for an adversary that outputs a random bit:

$$\mathbf{Adv}_F^{FG}(\mathcal{A}) = Pr[\mathbf{Expt}_F^{FG}(\mathcal{A}) = 1] - \frac{1}{2}$$

We then use Cryptoverif [Bla07][1], an automated prover of cryptographic properties in the computational model, to get an upper bound on this advantage.

[1] `http://prosecco.gforge.inria.fr/personal/bblanche/cryptoverif/`

Advantage and bounds

For all adversaries $\mathcal{A}$ making q_G queries to *Gen* and q_H queries to the hash function, there exists an adversary $\mathcal{B}$ (making $q_G + 1$ queries to $\mathcal{O}$ in its game) such that:

$$\mathbf{Adv}_F^{FG}(\mathcal{A}) \leq \frac{2q_G^2 + 2q_G}{2^{\eta_c}} + \frac{2q_G^2 + 4q_G + 2}{2^{\eta_n}} + 2\mathbf{Adv}_F^{PRP-CPA}(\mathcal{B})$$

Advantage and bounds

For all adversaries $\mathcal{A}$ making q_G queries to *Gen* and q_H queries to the hash function, there exists an adversary $\mathcal{B}$ (making $q_G + 1$ queries to $\mathcal{O}$ in its game) such that:

$$\mathbf{Adv}_F^{FG}(\mathcal{A}) \leq \frac{2q_G^2 + 2q_G}{2^{\eta_c}} + \frac{2q_G^2 + 4q_G + 2}{2^{\eta_n}} + 2\mathbf{Adv}_F^{PRP-CPA}(\mathcal{B})$$

From this, we can choose our parameters to achieve the desired security bound. For instance, if we use a $\eta_c = 128$b block cipher and a $\eta_n = 96$b nonce, and we allow the attacker $q_G = 2^{20}$ oracle calls and answers, we get:

$$\mathbf{Adv}_F^{FG}(\mathcal{A}) \leq \frac{2 \times (2^{20})^2 + 2 \times 2^{20}}{2^{128}} + \frac{2 \times (2^{20})^2 + 4 \times 2^{20} + 2}{2^{96}} + 2\mathbf{Adv}_F^{PRP-CPA}(\mathcal{B})$$

$$= ... = 2^{-54.999} + 2\mathbf{Adv}_F^{PRP-CPA}(\mathcal{B})$$

Choosing the block cipher and final bound

$$\mathbf{Adv}_F^{FG}(\mathcal{A}) \leq 2^{-54.999} + 2\mathbf{Adv}_F^{PRP-CPA}(\mathcal{B})$$

If we use AES-128 as block cipher, the best attack known to this day needs $2^{126.1}$ operations. Therefore, we can expect $\mathbf{Adv}_{AES-128}^{PRP-CPA}(\mathcal{B})$ to be much smaller than 2^{-54} for any adversary $\mathcal{B}$.

Choosing the block cipher and final bound

$$\mathbf{Adv}_F^{FG}(\mathcal{A}) \leq 2^{-54.999} + 2\mathbf{Adv}_F^{PRP-CPA}(\mathcal{B})$$

If we use AES-128 as block cipher, the best attack known to this day needs $2^{126.1}$ operations. Therefore, we can expect $\mathbf{Adv}_{AES-128}^{PRP-CPA}(\mathcal{B})$ to be much smaller than 2^{-54} for any adversary $\mathcal{B}$.

- Using nonces of 12 bytes and hashes of 8 bytes,
- Using a correct blockcipher of 16 bytes (*e.g.* *AES*-128),
- With an attacker that has 2^{20} packets with chosen data, hash function calls, and validity queries,

The probability of the adversary breaking one of the three properties is less than 2^{-50}.

Outline

Comparison points

Protocols with different underlying mechanisms

- Uniform Random Walk
- Gradient Based Routing (GBR) plus variants built for enhanced resiliency [EOKMV11]
  - RGBR (randomly select a lower height neighbor each time)
  - PRGBR (send to the same height with $p = 0.4$)
  - PRDGBR (PRGBR + duplication at each node)
- Greedy-Face-Greedy (geographical)

Simulation parameters, models and topologies

- Sinalgo[2], from ETH Zurich
- Asynchronous network (FIFO links)
- Every honest node generates messages, random intervals
- From 50 to 400 randomly positioned sensors, one centered sink
- Topology: Unit Disk Graphs, average degree from 8 to 32
- Connectivity: connected graphs only
- Overall, around 15k simulations, sending over 6 billion messages.

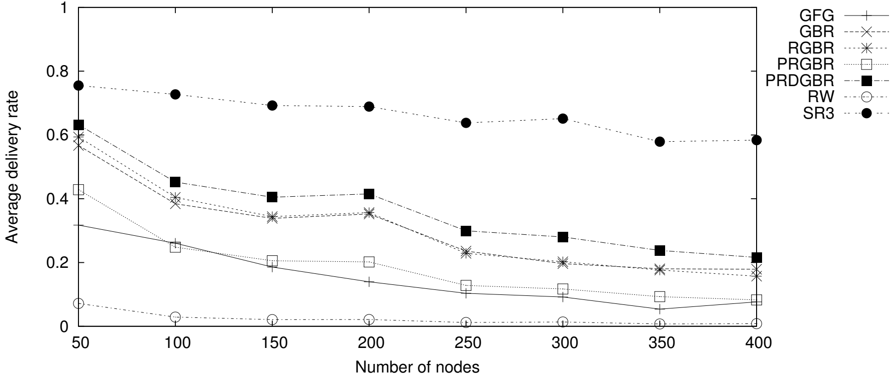
[2] `http://www.disco.ethz.ch/projects/sinalgo/`

Chosen attacks

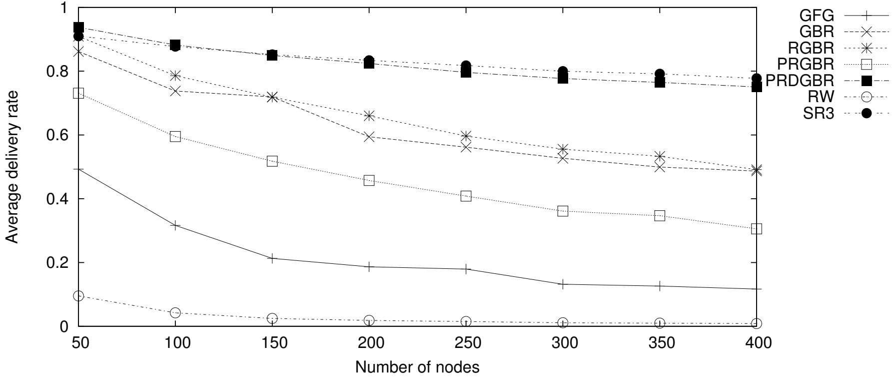
- Safe networks
- Blackholes (10%, 20% and 30%)
- Selective forwarding (10% and more, overall less impact than blackholes)
  - From 0 to 100% of messages transmitted
- 5% Wormholes→Blackholes + 5% Blackholes
  - Direct link to the sink, switch behaviors at one third of the simulation
- Sybil nodes (10% and more, not significantly better than blackholes)
  - From 1 to 5 identities

All attackers are randomly positioned.

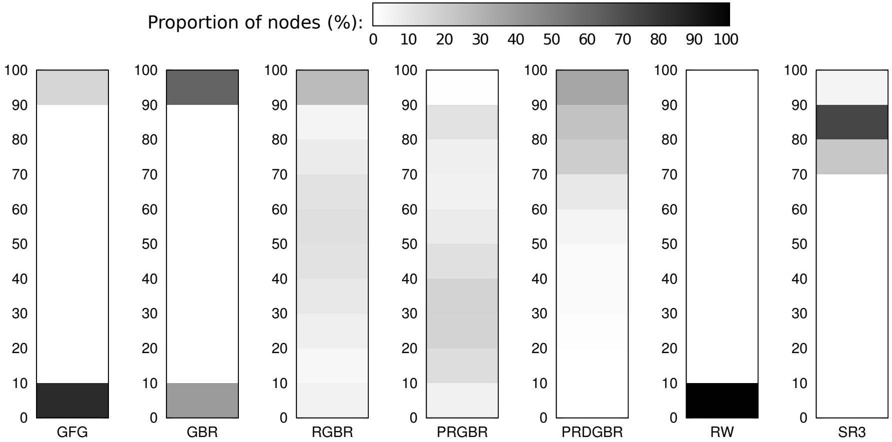
30% Blackholes, degree 8



30% Blackholes, degree 32

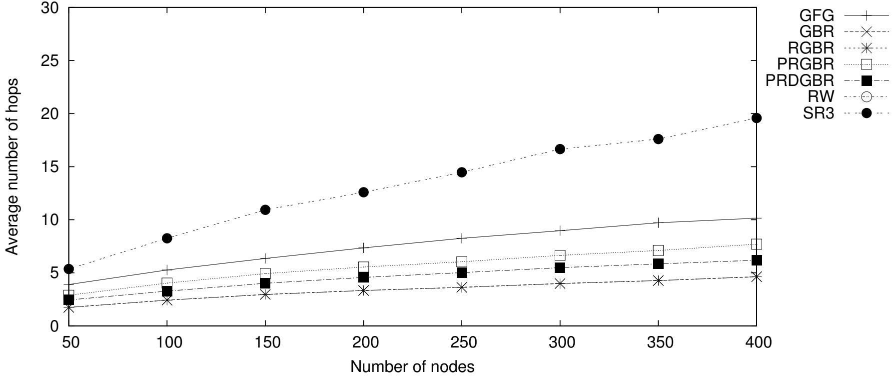


30% Blackholes, degree 32, 200 nodes



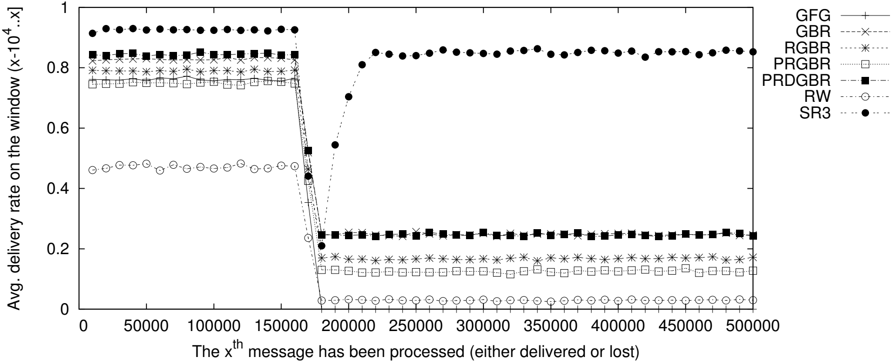
Distribution of the delivery rate (in %) for each node

No attackers, degree 16



For 400 nodes, RW is around 600 hops.

5% WH->BH, 5% BH, 200 nodes, degree 8, over time



Outline

Conclusion and future work

- SR3, Secure Reputation-based Resilient Routing
- Proved secure packet format using CryptoVerif
- Resilient against multiple attacks
- Efficient algorithm on both small (<50 nodes) and large (>400 nodes) networks

Future works:

- Dynamicity and mobility of nodes
- Implementation and energy measurements planned in ARESA2

Thanks for your attention.

Questions ?

rjamet@imag.fr

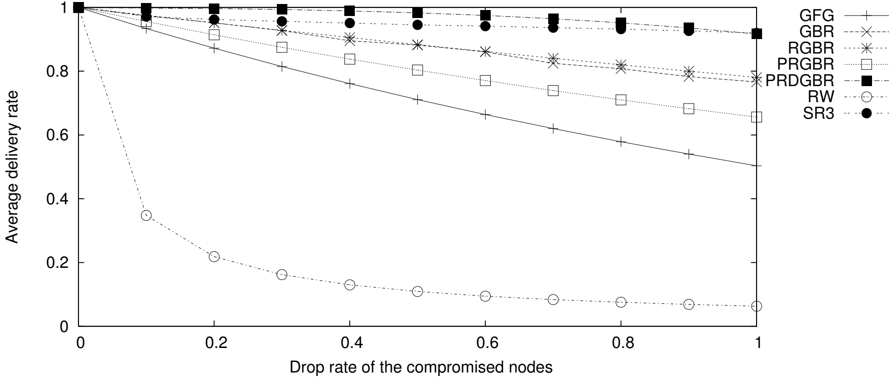
http://www-verimag.imag.fr/~rjamet/SR3/

Bibliography

B. Blanchet, *Cryptoverif: Computationally sound mechanized prover for cryptographic protocols*, Dagstuhl seminar "Formal Protocol Verification Applied, 2007, p. 117.

O. Erdene-Ochir, A. Kountouris, M. Minier, and F. Valois, *Enhancing resiliency against routing layer attacks in wireless sensor networks: Gradient-based routing in focus*, International Journal On Advances in Networks and Services 4 (2011), no. 1 and 2, 38–54.

20% Selective Forwarding, 200 nodes, degree 8, varying p



10% Sybil, 200 nodes, degree 8, varying number of identities