Security and Privacy in Wireless Networks

Mohammad Hossein Manshaei

manshaei@gmail.com

Chapter 7: (secowinet.epfl.ch)

Ad Hoc Network Routing Protocols, Attacks on Routing, Countermeasures, Secured Routing, Routing Security in Sensor Networks

SECURE ROUTING IN MULTI-HOP WIRELESS NETWORKS

Chapter Outline

Ad hoc Routing Protocols: Classification

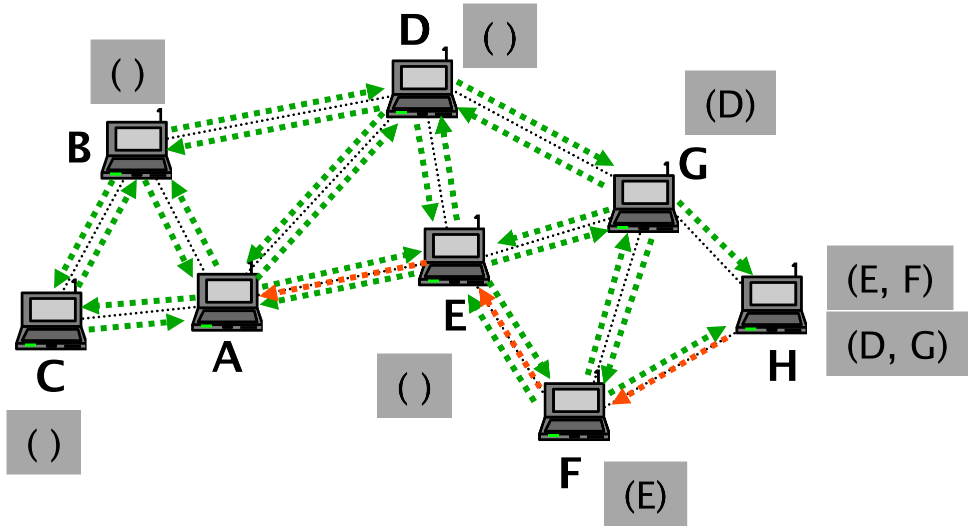
- **Topology-based protocols**
  - Proactive (Always up-to-date routing information)
    - distance vector based (e.g., DSDV)
    - link-state (e.g., OLSR)
  - Reactive (on-demand)
    - distance vector based (e.g., AODV)
    - source routing (e.g., DSR)
- **Position-based protocols**
    - greedy forwarding (e.g., GPSR, GOAFR)
    - restricted directional flooding (e.g., DREAM, LAR)
- **Hybrid approaches**

Example: Dynamic Source Routing (DSR)

- On-demand source routing protocol

- Two components:
  - route discovery
    - used only when source S attempts to send a packet to destination D
    - based on flooding of Route Requests (RREQ) and returning Route Replies (RREP)
  - route maintenance
    - makes S able to detect route errors (e.g., if a link along that route no longer works)

DSR Route Discovery illustrated



A → *: [RREQ, id, A, H; ()]
B → *: [RREQ, id, A, H; (B)]
C → *: [RREQ, id, A, H; (C)]
D → *: [RREQ, id, A, H; (D)]
E → *: [RREQ, id, A, H; (E)]
F → *: [RREQ, id, A, H; (E, F)]
G → *: [RREQ, id, A, H; (D,G)]

H → A: [RREP, <source route>; (E, F)]

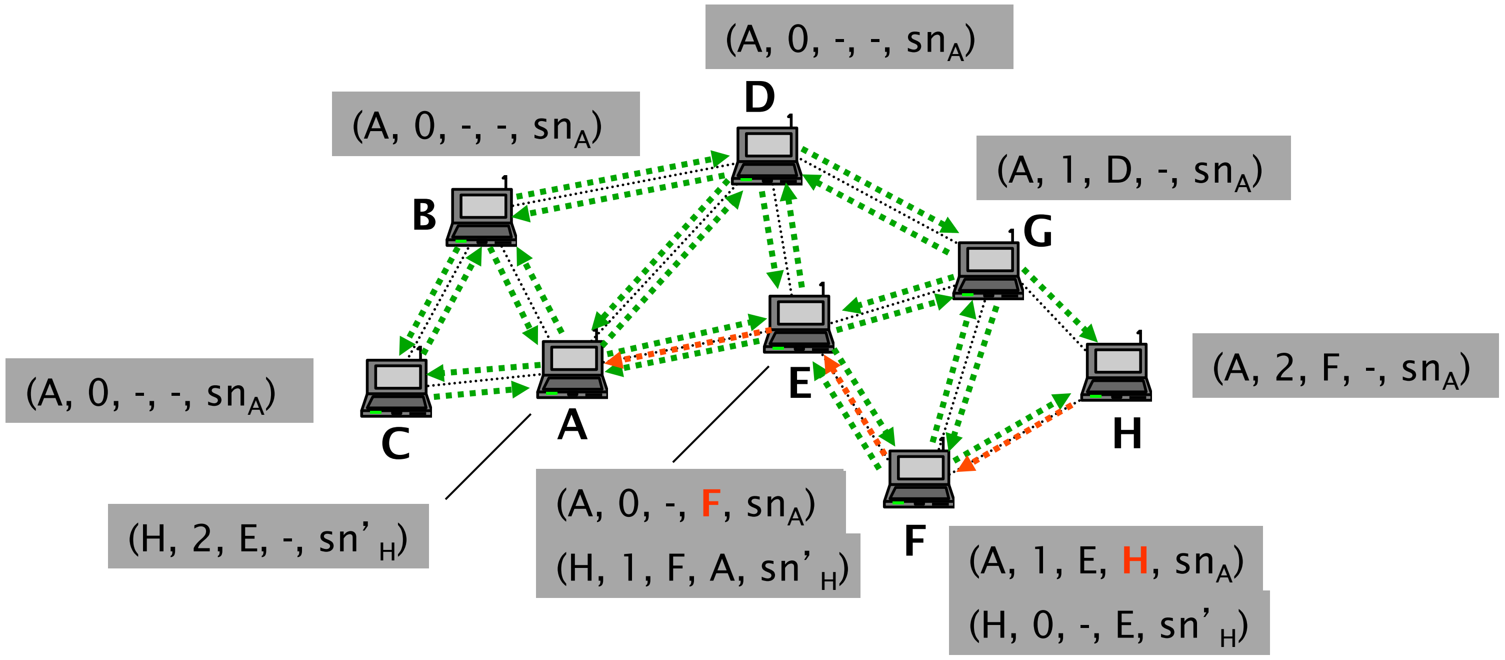
where <source route> is obtained

1. From the route cache of H
2. By reversing the route received in the RREQ
   - ➢ works only if all the links along the discovered route are bidirectional
   - ➢ IEEE 802.11 assumes that links are bidirectional
3. by executing a route discovery from H to A
   - ➢ discovered route from A to H is piggy backed to avoid infinite recursion

Example: Ad-hoc On-demand Distance Vector (AODV) Routing

- On-demand distance vector routing
- Uses sequence numbers to ensure loop-freedom and to detect out-of-date routing information
- Operation is similar to that of DSR **but the nodes maintain routing tables** instead of route caches
- **A routing table entry contains the following:**
  - destination identifier
  - number of hops needed to reach the destination
  - identifier of the next hop towards the destination
  - list of precursor nodes (that may forward packets to the destination via this node)
  - destination sequence number

AODV Route Discovery illustrated



A → *: [RREQ, id, A, H, 0, sn_A, sn_H]
B → *: [RREQ, id, A, H, 1, sn_A, sn_H]
C → *: [RREQ, id, A, H, 1, sn_A, sn_H]
D → *: [RREQ, id, A, H, 1, sn_A, sn_H]
E → *: [RREQ, id, A, H, 1, sn_A, sn_H]
F → *: [RREQ, id, A, H, 2, sn_A, sn_H]
G → *: [RREQ, id, A, H, 2, sn_A, sn_H]

H → F: [RREP, A, H, 0, sn'_H]
F → E: [RREP, A, H, 1, sn'_H]
E → A: [RREP, A, H, 2, sn'_H]

Proactive Routing

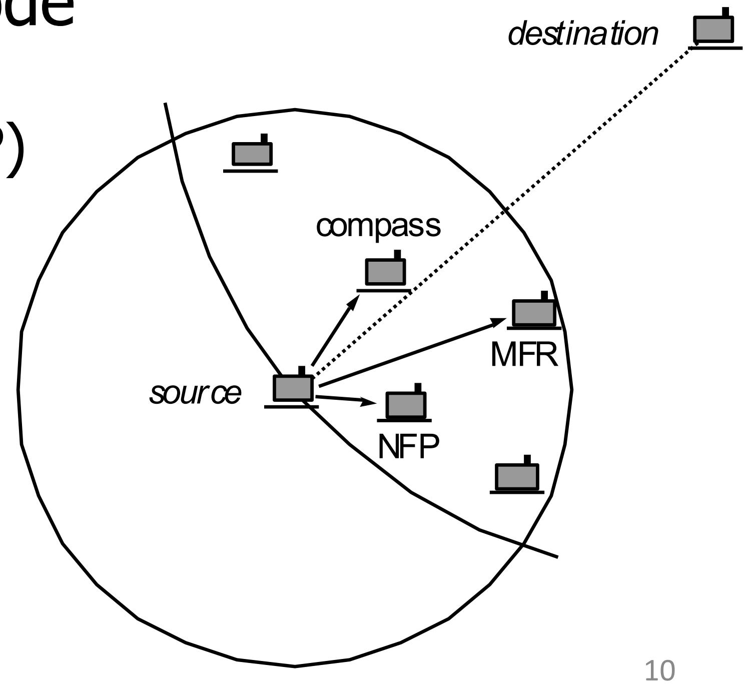
1. Link-State Protocols
   - Each node periodically floods the network with a message that contains the state of the links of that node (OLSR in MANET)
2. Distance Vector Protocols
   - Nodes execute a distributed shortest path algorithm to determine the best route to every other node in the network (DSDV in MANET)

Example: Position-based Greedy Forwarding

- **Assumptions**
  - nodes are aware of their own positions and that of their neighbors
  - packet header contains the position of the destination

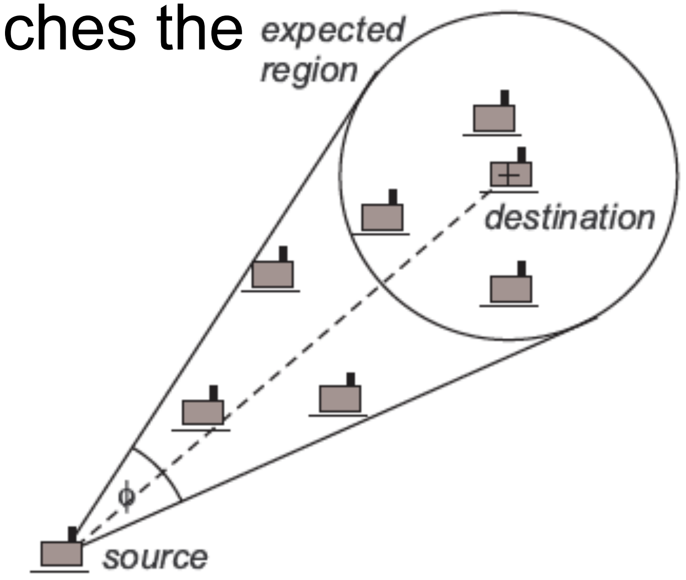
- Packet is forwarded to a neighbor that is closer to the destination than the forwarding node
  - Most Forward within Radius (MFR)
  - Nearest with Forward Progress (NFP)
  - Compass forwarding
  - Random forwarding

- **Additional mechanisms are needed to cope with local minimums (dead-ends)**



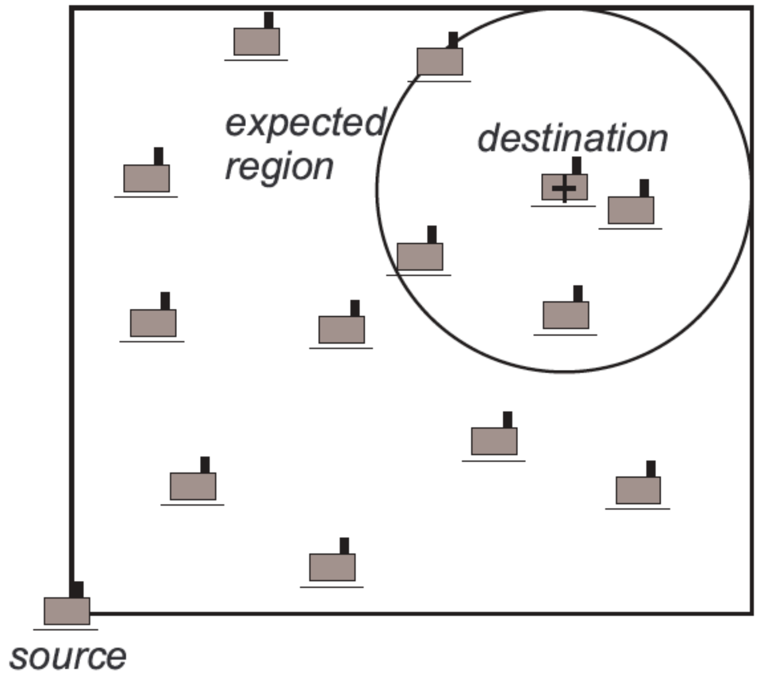
DREAM (Distance Routing Effect Algorithm for Mobility)

- An expected region of the destination is calculated.
- The direction to the destination is defined by the line between the forwarding node and the center of the destination's expected region, and the angle *Φ*.
- Each neighbor of the forwarding node that lies within this angle must re-broadcast the packet.
- These calculations are repeated by each intermediate node that receives the packet until it reaches the destination.



LAR (Location Aided Routing)

- The source of the data packet calculates an expected region of the destination, and then the packet is flooded within the rectangular region.
- Nodes outside this region will drop packets



Chapter Outline

Attacks on Routing Protocols (1/2)

- **General objectives of attacks**
  1. increase adversarial control over the communications between some nodes;
  2. degrade the quality of the service provided by the network;
  3. increase the resource consumption of some nodes (e.g., CPU, memory, or energy).

- **Adversary model**
  – Insider adversary
    - can corrupt legitimate nodes
  – The attacker is not all-powerful
    - it is not physically present everywhere
    - it launches attacks from regular devices

Attacks on Routing Protocols (2/2)

- **Attack mechanisms**
  - eavesdropping, replaying, modifying, and deleting control packets
  - fabricating control packets containing fake routing information (forgery)
  - fabricating control packets under a fake identity (spoofing)
  - dropping data packets (attack against the forwarding function)
  - wormholes and tunneling (*These are different!*)
  - rushing

- **Types of attacks**
  - route disruption
  - route diversion
  - creation of incorrect routing state
  - generation of extra control traffic
  - creation of a gray hole

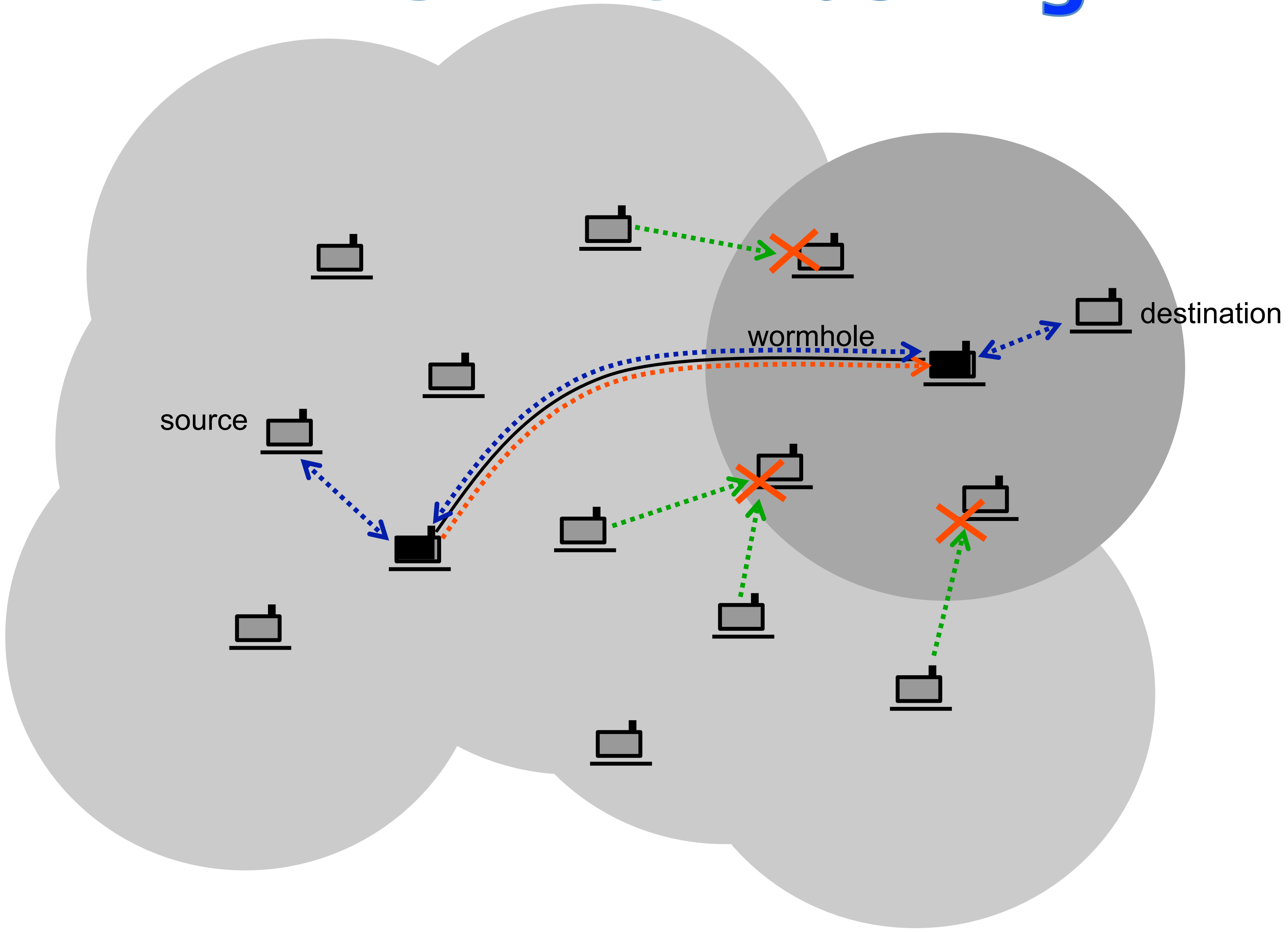
Route Disruption

- The adversary prevents a route from being discovered between two nodes that are otherwise connected

- The primary objective of this attack is to degrade the quality of service provided by the network
  - the two victims cannot communicate, and
  - other nodes can also suffer and be coerced to use suboptimal routes

- Attack mechanisms that can be used to mount this attack:
  - dropping route request or route reply messages on a vertex cut
  - forging route error messages
  - combining wormhole/tunneling and control packet dropping
  - rushing

Example: Route Disruption in DSR with Rushing



Route Diversion

- Due to the presence of the adversary, the protocol establishes routes that are different from those that it would establish, if the adversary did not interfere with the execution of the protocol

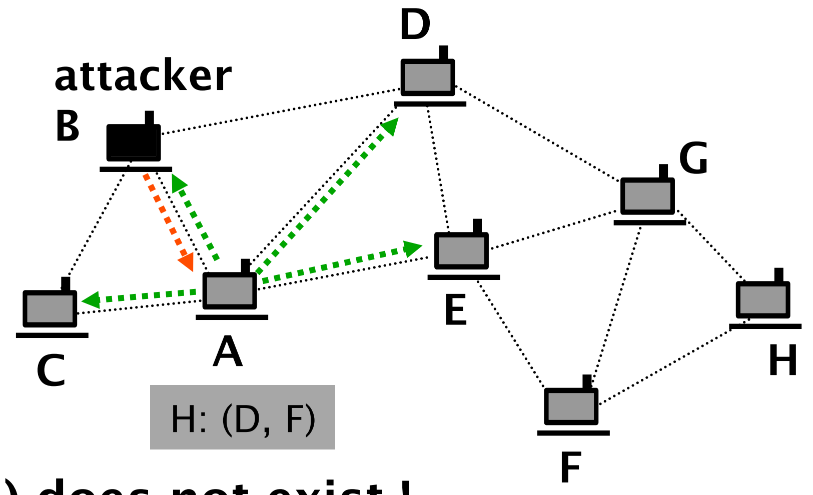
- The objective of route diversion can be
  - to increase adversarial control over the communications between some victim nodes
    - the adversary tries to achieve that the diverted routes contain one of the nodes that it controls or a link that it can observe
    - the adversary can eavesdrop or modify data sent between the victim nodes easier
  - to increase the resource consumption of some nodes
    - many routes are diverted towards a victim that becomes overloaded
  - degrade quality of service
    - by increasing the length of the discovered routes, and thereby, increasing the end-to-end delay between some nodes

- Route diversion can be achieved by
  - forging or manipulating routing control messages
  - dropping routing control messages
  - setting up a wormhole/tunnel

Creation of Incorrect Routing State

- This attack aims at jeopardizing the routing state in some nodes so that the state appears to be correct but, in fact, it is not
  - data packets routed using that state will never reach their destinations
- The objective of creating incorrect routing state is
  - to increase the resource consumption of some nodes
    - the victims will use their incorrect state to forward data packets, until they learn that something goes wrong
  - to degrade the quality of service
- Can be achieved by
  - spoofing, forging, modifying, or dropping control packets

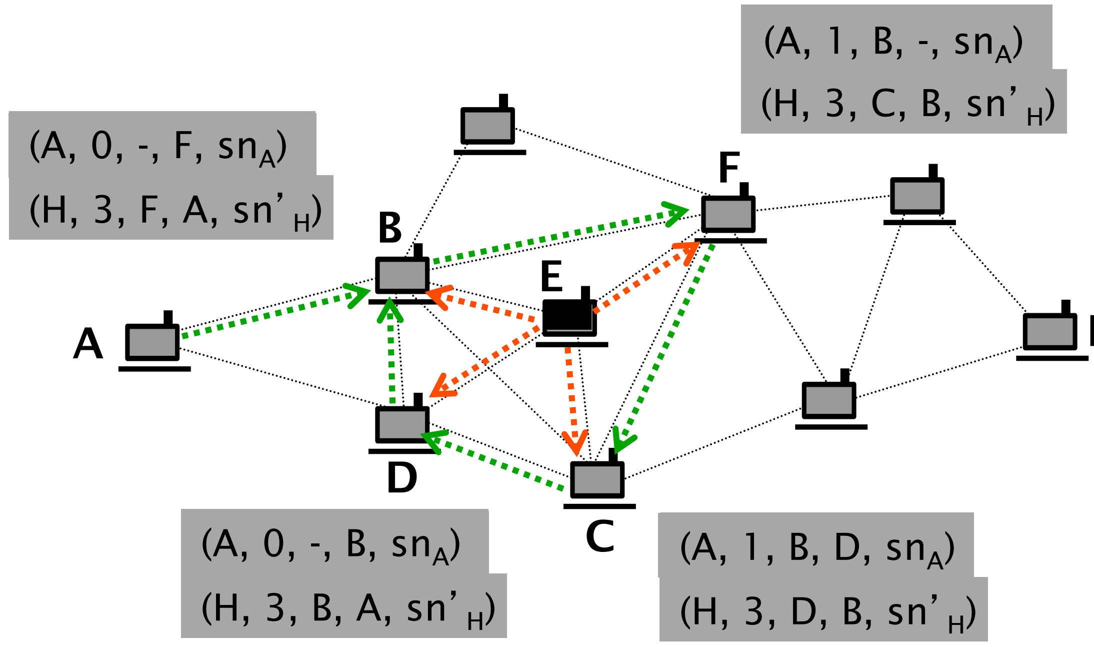
Example: Creation of Incorrect Routing State in DSR



Route (A, D, F, H) does not exist !

A → *: [RREQ, id, A, H; ()]
B → A: [RREP, <src route>, A, H; (D, F)]

Example: Creation of Incorrect Routing State in AODV



E (C) → F: [RREP, A, H, 2, sn'_H]
E (D)→ C: [RREP, A, H, 2, sn'_H]
E (B)→ D: [RREP, A, H, 2, sn'_H]
E (F)→ B: [RREP, A, H, 2, sn'_H]

Generation of Extra Control Traffic

- Injecting spoofed control packets into the network
- Aiming at increasing resource consumption due to the fact that such control packets are often flooded in the entire network

Setting up a Gray Hole

- An adversarial node selectively drops data packets that it should forward

- **The objective is**
  - to degrade the quality of service
    - packet delivery ratio between some nodes can decrease considerably
  - to increase resource consumption
    - wasting the resources of those nodes that forward the data packets that are finally dropped by the adversary

- **Implementation is trivial**
  - adversarial node participates in the route establishment
  - when it receives data packets for forwarding, it drops them
  - even better if combined with wormhole/tunneling

Chapter Outline

Countermeasures

- Authentication of control packets
  – using MACs or digital signatures
- Protection of mutable information in control packets
  – using MACs or digital signatures
  – often complemented with the use of one-way hash functions
- Detecting wormholes and tunnels
- Combating gray holes
  – using multi-path routing
  – using a “detect and react” approach

Authentication of control packets

- **Questions:**
  - Who should authenticate the control packets?
  - Who should be able to verify authenticity?
- **Control packets should be authenticated by their originators**
- **Authenticity should be verifiable by the target of the control packet**
- Moreover, **each node that updates its routing state as a result of processing the control packet must be able to verify its authenticity**
  - the adversary can still mount resource consumption attacks
- Each node that processes and re-broadcasts or forwards the control packet must be able to verify its authenticity
- As it is not known in advance which nodes will process a given control packet, we need a **broadcast authentication** scheme

Protection of mutable information in control packets

- Often, intermediate nodes add information to the control packet before re-broadcasting or forwarding it (hop count, node list, etc.)
- This added information is not protected by control packet origin authentication
- ***Each node that adds information to the packet should authenticate that information in such a way that each node that acts upon that information can verify its authenticity***
- This works for traceable additions (e.g., adding node identifiers), but what about untraceable additions (e.g., increasing the hop count)?

Protection of traceable modifications

- The entire control packet can be re-signed by each node that modifies it

- **Problems**:
  - signatures can be removed from the end
    - one-way hash chains can be used (e.g., Ariadne)
    - efficient aggregate signatures provide better solution
  - re-signing increases the resource consumption of the nodes (potentially each node needs to re-sign broadcast messages)
    - no easy way to overcome this problem
    - one approach is to avoid mutable information in control packets
    - another approach is to sacrifice some amount of security (e.g., SRP)
  - corrupted nodes can still add incorrect information and sign it
    - very tough problem …

Protection of untraceable modifications

- No perfect solution exists (trust problem)

- Hop counts are often protected by a per-hop hashing mechanism (e.g., SAODV, SEAD)
  - control packets contain a hash value associated with the hop-count
  - when the control packet is forwarded or re-broadcast, the hop-count is incremented and the hash value is hashed once
  - adversarial nodes cannot decrease hop-count values in control packets because that would need to compute pre-images of hash values
  - adversary can still increase the hop-count …

- Another approach is to eliminate hop-counts
  - use other routing metrics (e.g., ARAN uses the delay as the routing metric)

Combating Gray Holes

1. use multiple, preferably disjoint routes
   - increased robustness
   - but also increased resource consumption
   - resource consumption can be somewhat decreased by applying the principles of error correcting coding
     - data packet is coded and the coded packet is split into smaller chunks
     - a threshold number of chunks is sufficient to reconstruct the entire packet
     - chunks are sent over different routes

2. detect and react
   - monitor neighbors and identify misbehaving nodes
   - use routes that avoid those misbehaving nodes
   - reputation reports about nodes can be spread in the network
   - this approach has several problems
     - how to detect reliably that a node is misbehaving?
     - how to prevent false accusations and spreading of negative reputations?

Some secure ad hoc network routing protocols

- SRP (on-demand source routing)
- Ariadne (on-demand source routing)
- endairA (on-demand source routing)
- S-AODV (on-demand distance vector routing)
- ARAN (on-demand, routing metric is the propagation delay)
- SEAD (proactive distance vector routing)
- SMT (multi-path routing combined error correcting)
- Watchdog and Pathrater (implementation of the “detect and react” approach to defend against gray holes)
- ODSBR (source routing with gray hole detection)

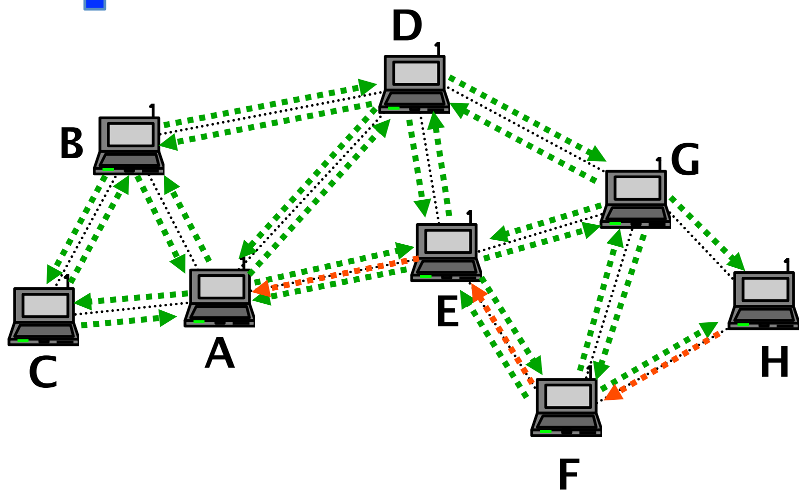
SRP (Secure Routing Protocol)

- ➢ SRP is a secure variant of DSR

- ➢ uses symmetric-key authentication (MACs)
  - – due to mobility, it would be impractical to require that the source and the destination share keys with all intermediate nodes
  - – hence there's only a shared key between the source and the destination
  - →only end-to-end authentication is possible
  - →no optimizations

- ➢ SRP is simple but it does not prevent the manipulation of mutable information added by intermediate nodes
  - – this opens the door for some attacks
  - – some of those attacks can be thwarted by secure neighbor discovery protocols

SRP Operation Illustrated



A → * : [RREQ, A, H, id, sn, mac_{AH}, ()]
B → * : [RREQ, A, H, id, sn, mac_{AH}, (B)]
C → * : [RREQ, A, H, id, sn, mac_{AH}, (C)]
D → * : [RREQ, A, H, id, sn, mac_{AH}, (D)]
E → * : [RREQ, A, H, id, sn, mac_{AH}, (E)]
F → * : [RREQ, A, H, id, sn, mac_{AH}, (E, F)]
G → * : [RREQ, A, H, id, sn, mac_{AH}, (D, G)]

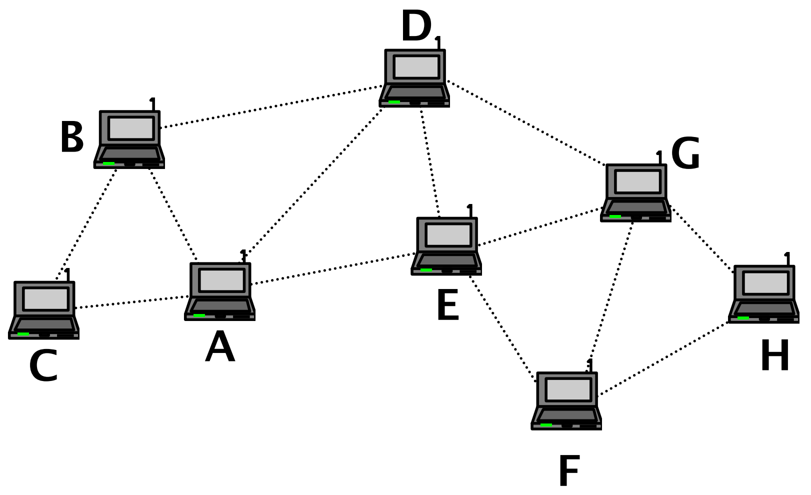
H → A : [RREP, A, H, id, sn, (E, F), mac_{HA}]

mac_{AH}: Message Authentication Code covering RREQ, A, H, id, and sn

Ariadne

- Ariadne is another secured variant of DSR
- it uses control message authentication to prevent modification and forgery of routing messages
  - based on signatures, MACs, or TESLA
- it uses a per-hop hash mechanism to prevent the manipulation of the accumulated route information in the route request message

Ariadne with signatures



A : h_A = mac_{AH}(RREQ | A | H | id)
A → * : [RREQ, A, H, id, h_A, (), ()]

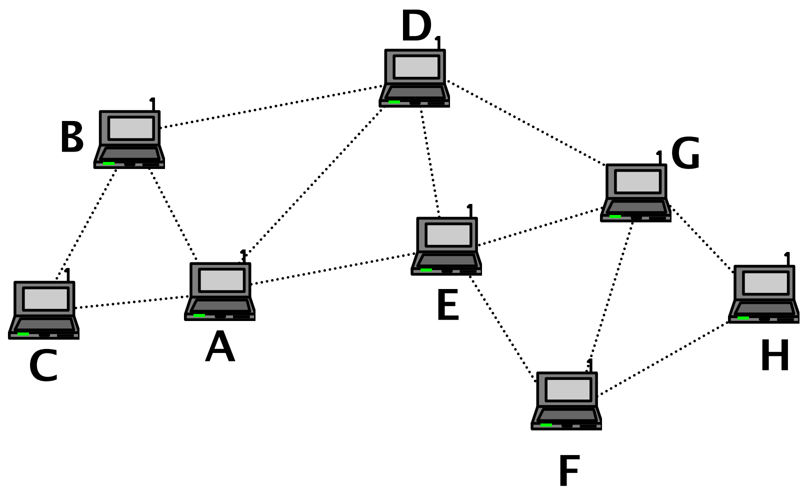
E : h_E = H(E | h_A)
E → * : [RREQ, A, H, id, h_E, (E), (sig_E)]

F : h_F = H(F | h_E)
F → * : [RREQ, A, H, id, h_F, (E, F), (sig_E, sig_F)]

H → A: [RREP, H, A, (E, F), (sig_E, sig_F), sig_H] (sent via F and E)

Each signature is computed over the message fields preceding it

Ariadne with standard MACs



A : $h_A = mac_{AH}($ RREQ | A | H | id)
A → * : [RREQ, A, H, id, h_A, (), ()]

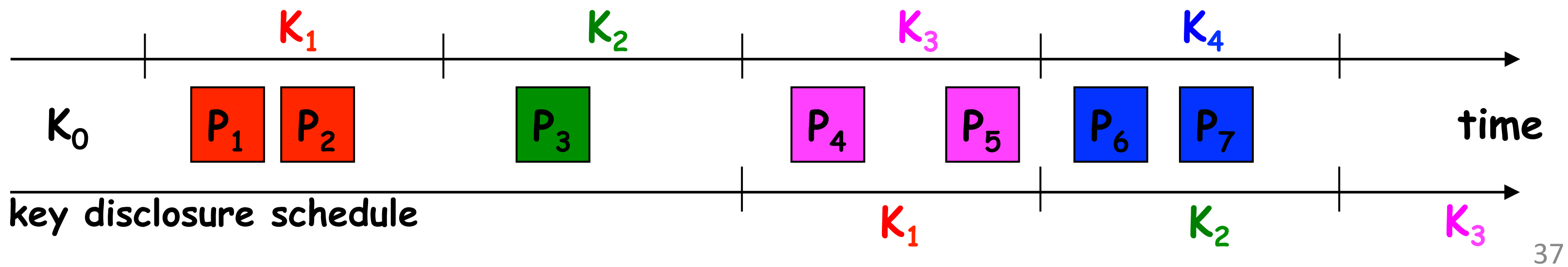
E : $h_E = H($ E | h_A)
E → * : [RREQ, A, H, id, h_E, (E), (mac_{EH})]

F : $h_F = H($F | h_E)
F → * : [RREQ, A, H, id, h_F, (E, F), (mac_{EH}, mac_{FH})]

H → A : [RREP, H, A, (E, F), mac_{HA}]

Symmetric-key broadcast authentication with TESLA

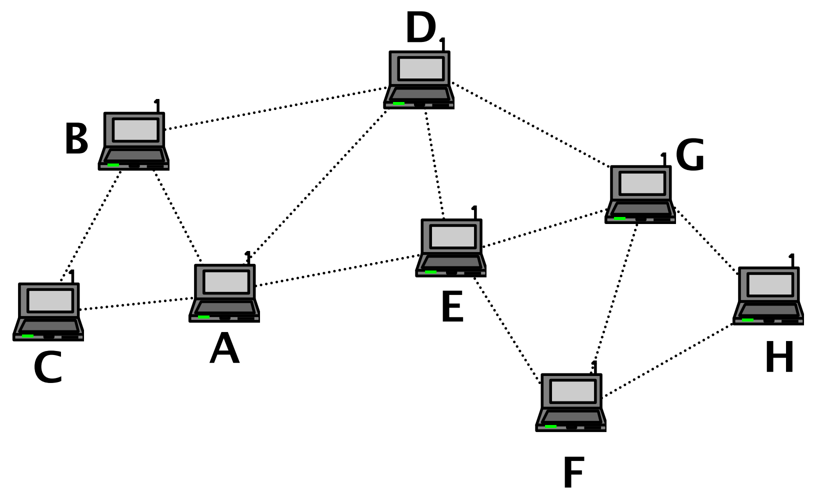
- MAC keys are consecutive elements in a one-way key chain:
  - $K_n \rightarrow K_{n-1} \rightarrow \ldots \rightarrow K_0$
  - $K_i = h(K_{i+1})$
- TESLA protocol:
  - setup: K_0 is sent to each node in an authentic way
  - time is divided into epochs
  - each message sent in epoch i is authenticated with key K_i
  - K_i is disclosed in epoch i+d, where d is a system parameter
  - K_i is verified by checking $h(K_i) = K_{i-1}$
- example:



Ariadne with TESLA

- **Assumptions:**
  - each source-destination pair (S, D) shares a symmetric key K_{SD}
  - each node F has a TESLA key chain $K_{F,i}$
  - each node knows an authentic TESLA key of every other node
- **Route request (source S, destination D):**
  - S authenticates the request with a MAC using K_{SD}
  - each intermediate node F appends a MAC computed with its current TESLA key
  - D verifies the MAC of S
  - D verifies that the TESLA key used by F to generate its MAC has not been disclosed yet
- **Route reply:**
  - D generates a MAC using K_{SD}
  - each intermediate node delays the reply until it can disclose its TESLA key that was used to generate its MAC
  - F appends its TESLA key to the reply
  - S verifies the MAC of D, and all the MACs of the intermediate nodes

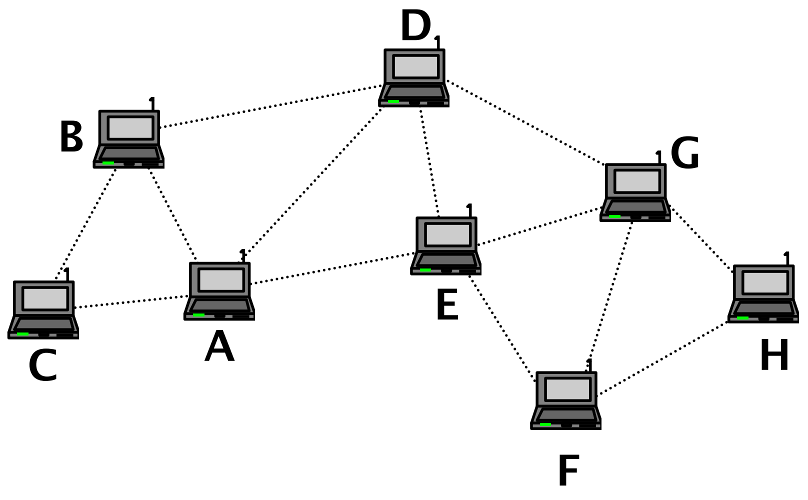
Ariadne with TESLA illustrated



A → *: [RREQ, A, H, id, h_A, (), ()]
E → *: [RREQ, A, H, id, h_E, (E), ($mac_{K_{E,i}}$)]
F → *: [RREQ, A, H, id, h_F, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$)]

H → F: [RREP, H, A, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$), mac_{HA}, ()]
F → E: [RREP, H, A, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$), mac_{HA}, ($K_{F,i}$)]
E → A: [RREP, H, A, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$), $mac_{K_{HA}}$, ($K_{F,i}$, $K_{E,i}$)]

endairA



target verifies:

- there's no repeating ID in the node list
- last node in the node list is a neighbor

each intermediate node verifies:

- its own ID is in the node list
- there's no repeating ID in the node list
- next and previous nodes in the node list are neighbors
- all signatures are valid

source verifies:

- there's no repeating ID in the node list
- first node in the node list is a neighbor
- all signatures are valid

A → * : [RREQ, A, H, id, ()]
E → * : [RREQ, A, H, id, (E)]
F → * : [RREQ, A, H, id, (E, F)]

H → F : [RREP, A, H, id, (E, F), (sig_H)]
F → E : [RREP, A, H, id, (E, F), (sig_H, sig_F)]
E → A : [RREP, A, H, id, (E, F), (sig_H, sig_F, sig_E)]

Properties of endairA

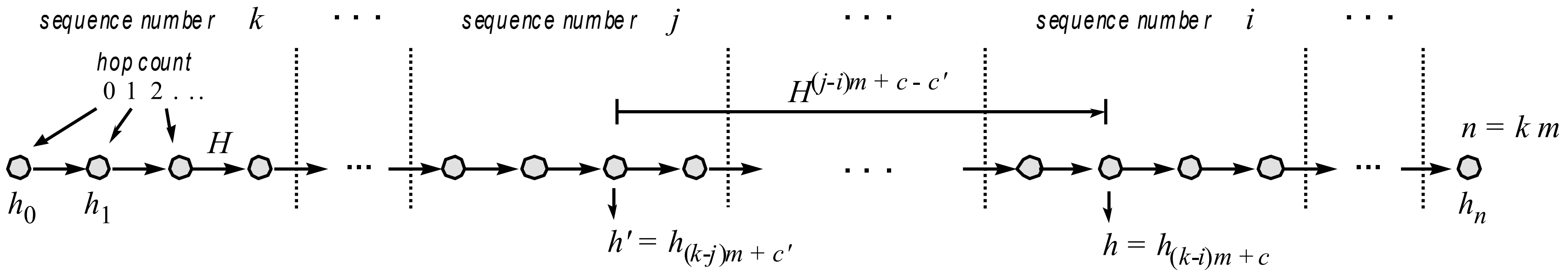
- security
  - endairA is provably secure if the signature scheme is secure against chosen message attacks
- efficiency
  - endairA requires less computation
    - route reply is signed and verified only by the nodes on the route
    - in Ariadne, route request is signed (and potentially verified) by every node in the network

SAODV (Secure AODV)

- SAODV is a secure variant of AODV
- protects non-mutable information with a digital signature (of the originator of the control packet)
- uses hash chains for the protection of the HopCount value
  - new non-mutable fields:
    - MaxHopCount (= TTL)
    - TopHash (= iterative hash of a random seed MaxHopCount times)
  - new mutable field:
    - Hash (contains the current hash value corresponding to the HopCount value)
- operation
  - initially Hash is set to the seed
  - each time a node increases HopCount, it also replaces Hash with H(Hash)
  - verification of the HopCount is done by hashing the Hash field MaxHopCount-HopCount times and checking if the result matches TopHash

SEAD (Secure Efficient Ad hoc Distance vector routing)

- SEAD is a proactive distance vector protocol
  - it can be viewed as a secure variant of DSDV
- SEAD tries to ensure that
  - sequence numbers cannot be increased
  - hop count values cannot be decreased
- operation
  - each node has a hash chain of length k times m (where m is the maximum diameter of the network)
  - when a node sends out a route update message about itself with sequence number i and hop count 0, it reveals $h_{(k-i)m}$
  - any node can increase the hop count by computing $h_{(k-i)m+c}$
  - any node can verify if the sequence number is greater than any previously known value

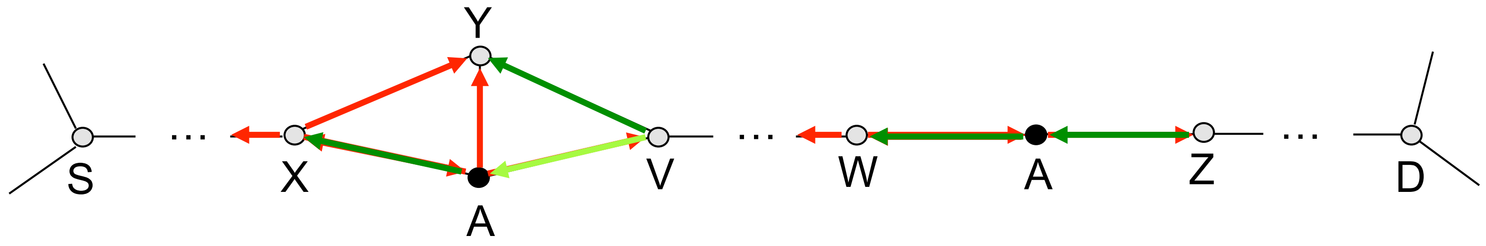


Chapter Outline

Provable security for ad hoc network routing protocols

- several “secure” routing protocols have been proposed for wireless ad hoc networks
  - SRP, Ariadne, SEAD, ARAN, S-AODV, …
- their security have been analyzed mainly by informal means
- informal reasoning about security protocols is prone to errors
  - lessons learnt in the field of key exchange protocols
  - some attacks have been found against Ariadne and S-AODV
- we need more assurances
  - mathematical models
  - precise definitions
  - sound proof techniques

An attack on Ariadne



X → * : [rreq, S, D, *id*, h_X, (…, X), (…, mac_{XD})]
A → * : [rreq, S, D, *id*, *, (…, X, A), (…, mac_{XD}, h_X)]
… …
W → * :[rreq, S, D, *id*, *, (…, X, A, V, …, W), (…, mac_{XD}, h_X, …, mac_{WD})]
A : h_A = H(A | h_X)
A → * : [rreq, S, D, *id*, h_A, (…, X, A), (…, mac_{XD}, mac_{AD})]
… …
Z → A : [rrep, D, S, (…, X, A, Z, …), mac_{DS}]
A → W : [rrep, D, S, (…, X, Y, V, … W, A, …), mac_{DS}]
… …
V → Y : [rrep, D, S, (…, X, Y, V, … W, A, …), mac_{DS}]
A → X : [rrep, D, S, (…, X, A, Z, …), mac_{DS}]
… …
? → S : [rrep, D, S, (…, X, A, Z, …), mac_{DS}] (a non-existent route!)

Mathematical framework

- based on the simulation paradigm
  - real-world model
    - describes the real operation of the protocol
  - ideal-world model
    - captures what the protocol wants to achieve in terms of security
  - definition of security in terms of indistinguishability of the two models from the point of view of honest participants

Mathematical framework (cont’d)

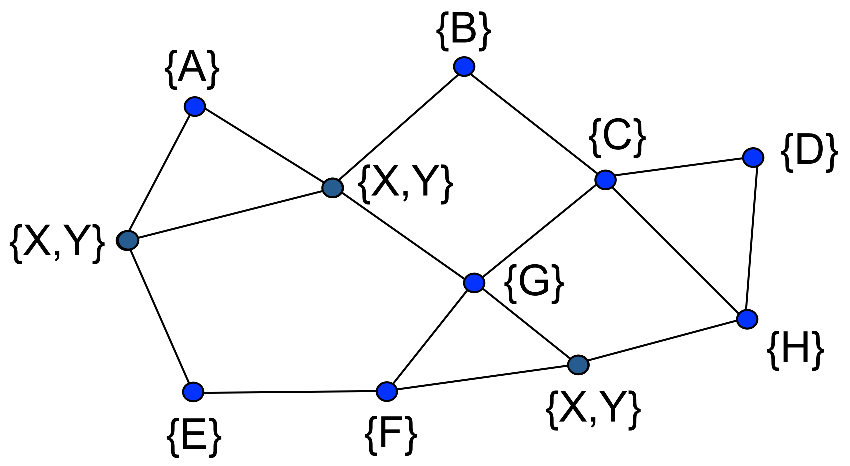
- communication model
  - multi-hop communication and the broadcast nature of radio channels are explicitly modeled
- adversary model
  - power of the adversary is limited
  - it has communication capabilities similar to regular nodes
  - it cannot fully control when the nodes receive messages
- model of computation
  - computation is not scheduled by the adversary
  - computation is performed in rounds (synchronous model)
  - knowledge of the current round number is never exploited
- ideal-world model and ideal-world adversary
  - they are essentially the same as the real-world model and adversary
  - the ideal world is ideal in the following sense:
    - route reply messages that contain incorrect routes are marked and filtered out
    - incorrect routes are never returned in the ideal world

Configuration

- an ad hoc network is represented by a graph $G(V, E)$
  - V: vertices are network nodes (honest and adversarial)
  - E: edges represent communication links (radio or wormhole)

- $V^* \subset V$ is a set of distinguished nodes (under the adversary's control)

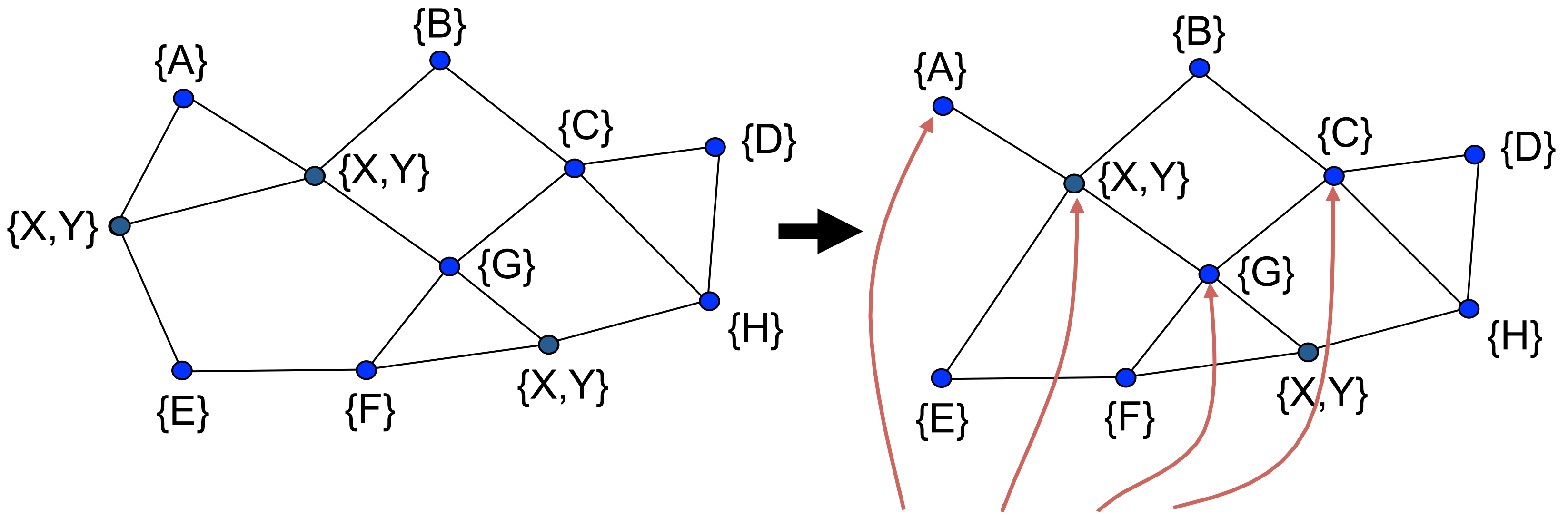
- $\mathcal{L}$ is a labeling function (assigns IDs to nodes) with the following restrictions:
  - each honest node has a unique, uncompromised ID
  - each adversarial node is labeled with *all* the compromised IDs
  - we assume that ID's are authenticated during neighbor discovery (Sybil attack is excluded)



- a *configuration* is a triplet: $(G, V^*, \mathcal{L})$

Plausible routes

- reduced configuration: $(\underline{G}(\underline{V}, \underline{E}), \underline{V}^*, \underline{\mathcal{L}})$
  - neighboring adversarial nodes are joined
- a route is *plausible* in a given configuration, if it doesn't contain repeating IDs and it can be partitioned in a way that each partition P can be associated with a node v in $\underline{G}$ such that
  - $P \subseteq \underline{\mathcal{L}}(v)$, and
  - neighboring partitions are associated with neighboring nodes in $\underline{G}$



A X Y G C → A | X Y | G | C

A X G D H → non-plausible

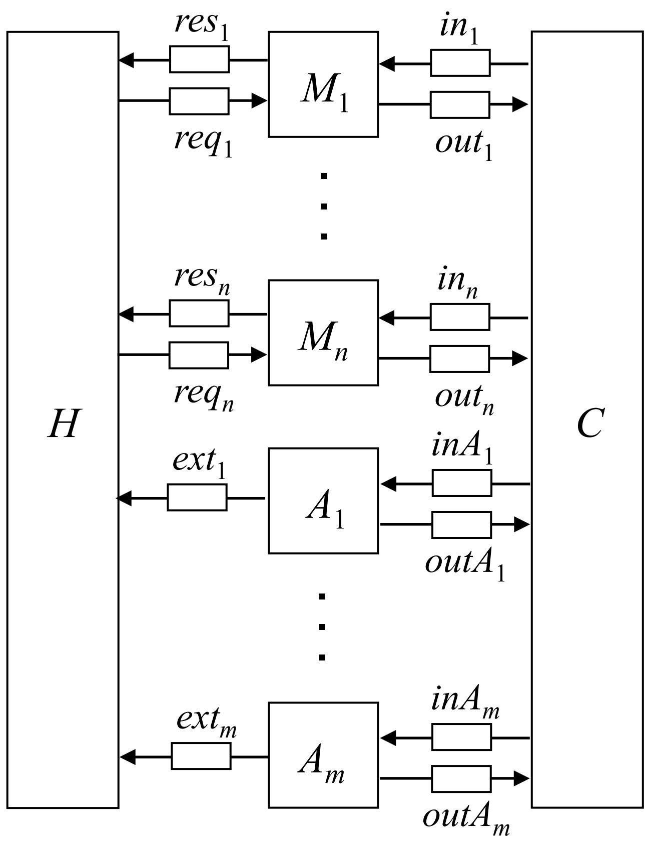
The rational behind plausible routes

- adversarial nodes can emulate the execution of the routing protocol (locally) using any subset of the compromised IDs in any order

- they can also pass information to each other in a proprietary way

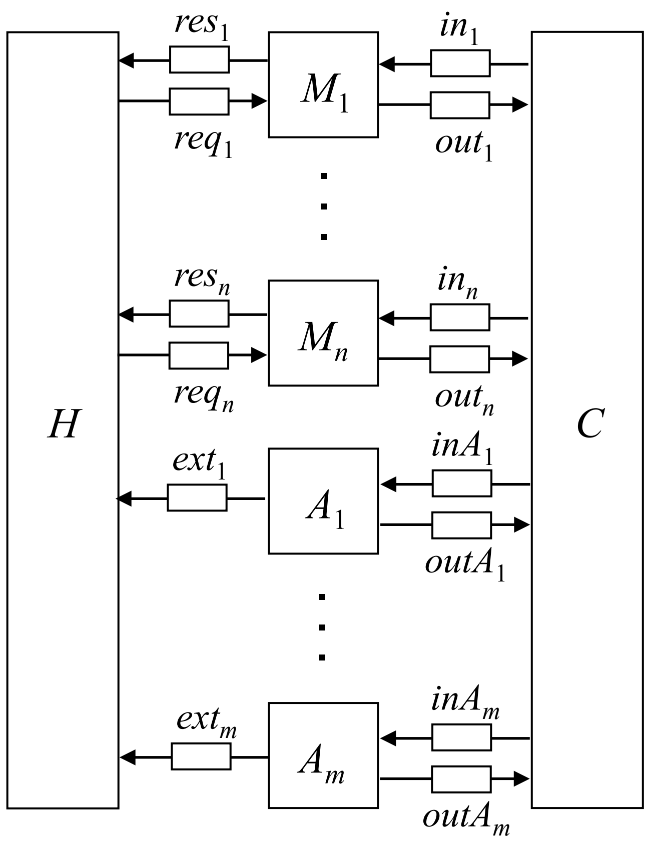
- these are *tolerable imperfections*, which are embedded in the notion of plausible routes

Real-world model (1)



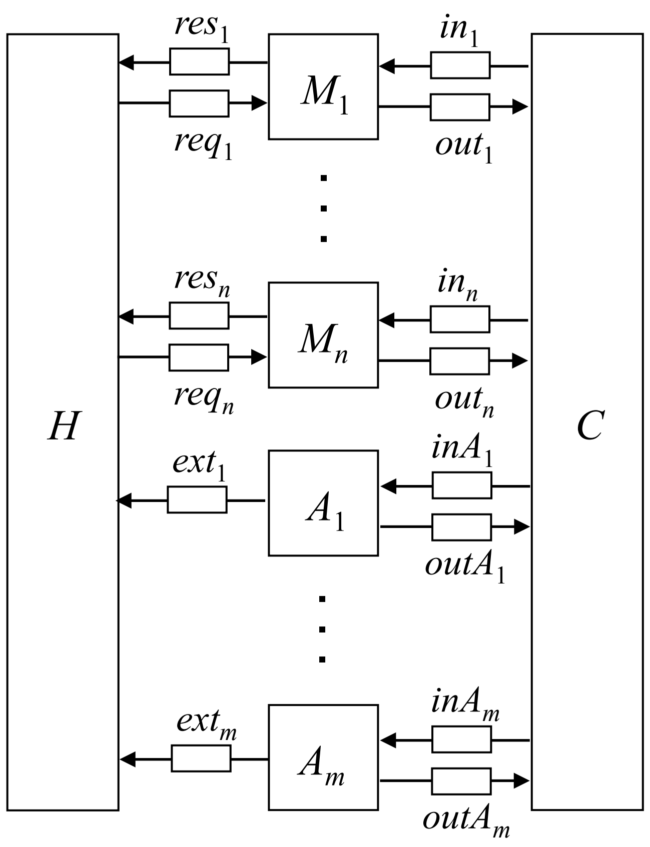
- $H, M_1, \ldots, M_n, A_1, \ldots, A_m, C$ are interacting, probabilistic Turing machines
  - $M_1, \ldots, M_n$ represent honest nodes in $\underline{G}$
  - $A_1, \ldots, A_m$ represent adversarial nodes in $\underline{G}$
  - C models the communication links (edges of $\underline{G}$)
- each machine is initialized with some input data (e.g., crypto keys) and some random input
- each machine operates in a reactive manner (must be activated)
  - reads input tape
  - performs state transition and writes output tape
  - goes back to sleep
- machines are activated by a hypothetic scheduler in rounds in a fix order in each round: $H, \ldots, C$
- the computation ends when H reaches a final state

Real-world model (2)



- C models the communication links
  - when activated, it moves the content of the output tape of each protocol machine (M_i and A_j) onto the input tape of all neighboring machines in $\underline{G}$ (in a random order)
- H models higher layer protocols (and ultimately the end-users) of non-corrupted nodes
  - it can initiate a route discovery process at any machine M_i by placing a request on req_i
  - a response may be returned to the request via res_i
  - the response contains a set of routes (maybe empty set)
  - it can receive out-of-band requests from the adversarial machines via ext_j

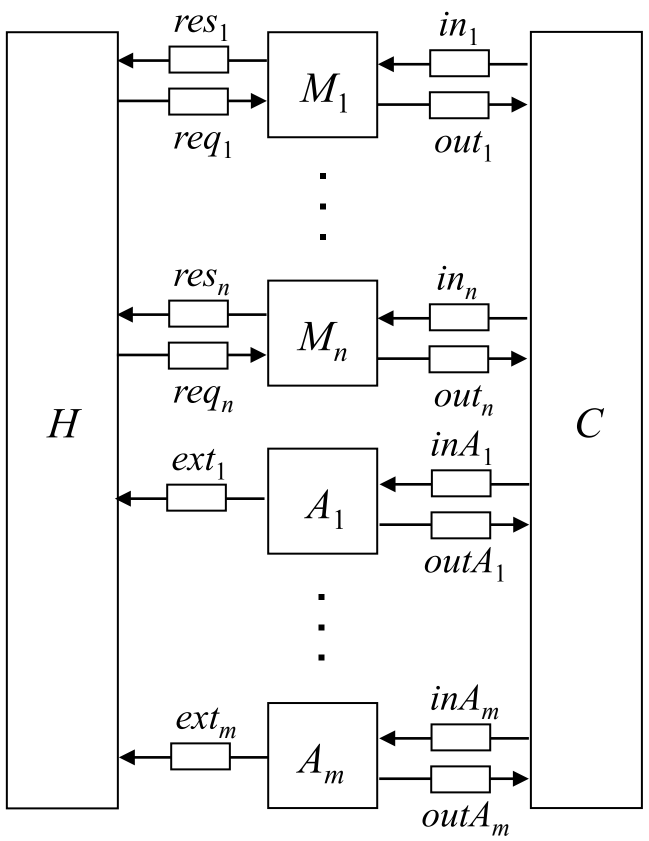
Real-world model (3)



- M_i models the operation of the routing algorithm in the i-th non-corrupted node
  - it receives requests from H via req_i and may return a response via res_i
  - it sends and receives routing messages to and from its neighbors via out_i and in_i
  - initialized with its own ID and those of its neighbors, some cryptographic material, and random input

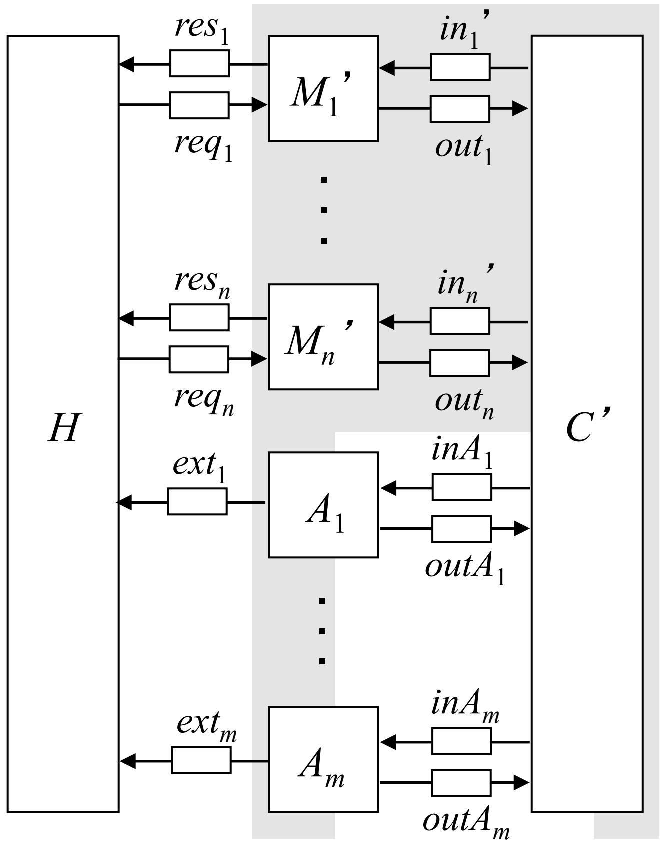
- A_j models the j-th adversarial node
  - it uses $outA_j$ and inA_j to communicate with its neighbors
  - it can use ext_j to “force” H to start a route discovery between any two *honest nodes*
  - it is *non-adaptive*: it places its requests on ext_j at the beginning of the computation, and doesn’t use ext_j anymore
  - its behavior is not restricted apart from being polynomial-time in the security parameter

Real-world model (4)



- output of the real-world model
  - sets of routes returned to H
  - denoted by $real\_out_{conf,\mathcal{A}}(r)$, where $r = (r_I, r_M, r_A, r_C)$
    - r_I – random input of cryptographic initialization (key generation)
    - r_M – random input of $M_1, \ldots, M_n$
    - r_A – random input of $A_1, \ldots, A_m$
    - r_C – random input of C
  - $real\_out_{conf,\mathcal{A}}$ denotes the random variable describing the output when r is chosen uniformly at random

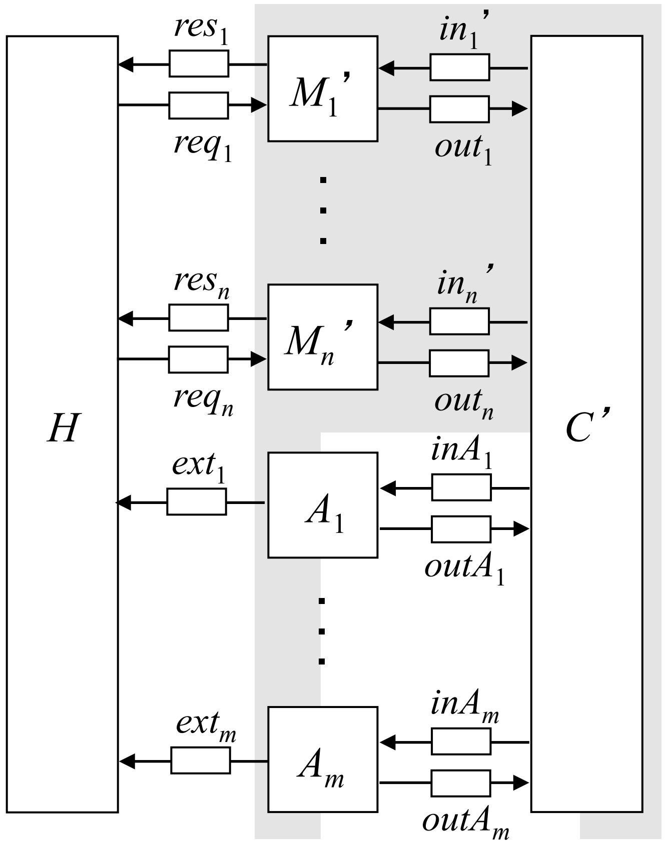
Ideal-world model (1)



- difference between C and C' :
  - C' marks every route reply message that contains a non-plausible route as corrupted before placing it on the input tape in_i' of a non-corrupted protocol machine M_i
  - otherwise C' works in the same way as C

- difference between M_i and M_i' :
  - when M_i' receives a route reply message that belongs to a route discovery process initiated by itself, it processes the message as follows:
    - it performs all the verifications required by the routing protocol
    - if the message passes all verifications, then it also checks the corruption flag attached to the message
    - if the message is corrupted (contains a non-plausible route), then M_i' drops the message
  - otherwise M_i' behaves as M_i

Ideal-world model (2)



- output of the ideal-world model
  - sets of routes returned to H
  - denoted by $ideal\_out_{conf,\mathcal{A}}(r')$, where $r' = (r'_I, r'_M, r'_A, r'_C)$
  - $ideal\_out_{conf,\mathcal{A}}$ denotes the random variable describing the output when r' is chosen uniformly at random

Definition of statistical security

A routing protocol is said to be statistically secure if, for any configuration $conf$ and any real-world adversary $\mathcal{A}$, there exists and ideal-world adversary $\mathcal{A}'$, such that

$$real\_out_{conf,\mathcal{A}} =^{s} ideal\_out_{conf,\mathcal{A}'}$$

where $=^{s}$ means statistically indistinguishable.

<u>notes</u>:

- two random variables are statistically indistinguishable if the L_1 distance of their distributions are negligibly small

- if Definition 1 is satisfied by a protocol, then a non-plausible route can be returned in the real system only with negligible probability (for every configuration and arbitrary adversary)

Proof technique

- let $\mathcal{A}' = \mathcal{A}$
- if, for a given r, no message is dropped due to its corruption flag in the ideal-world model, then the ideal-world model perfectly simulates the real-world model:

$$real\_out_{conf,\mathcal{A}}(r) = ideal\_out_{conf,\mathcal{A}}(r)$$

- if, for some r, there exist messages that are dropped due to their corruption flag in the ideal-world model, then there may be a *simulation failure*:

$$real\_out_{conf,\mathcal{A}}(r) \neq ideal\_out_{conf,\mathcal{A}}(r)$$

- in proofs, we want to show that simulation failures occur with negligible probability
- if this is not the case, then
  - in theory, we haven't proven anything (there may be another $\mathcal{A}' \neq \mathcal{A}$, for which we have statistical indistinguishability)
  - in practice, there's a problem with the protocol

Analysis of endairA (1)

Theorem:

endairA is statistically secure if the signature scheme is secure against chosen message attacks.

sketch of the proof:

- it is enough to prove that, for any configuration $conf$ and attacker $\mathcal{A}$, a route reply message in the ideal-world system is dropped due to its corruption flag set to true with negligible probability
- let us suppose that the following message is dropped due to its corruption flag:

  $$[\text{ rrep}, S, D, (N_1, N_2, \ldots, N_p), (sig_D, sig_{Np}, \ldots, sig_{N_1})\ ]$$
- we know that
  - there are no repeating IDs in $(S, N_1, N_2, \ldots, N_p, D)$
  - N_1 is a neighbor of S
  - all signatures are valid
  - S and D are honest
  - $(S, N_1, N_2, \ldots, N_p, D)$ is a non-plausible route in $\underline{G}$
- we prove that $\mathcal{A}$ must have forged a signature to achieve this

Analysis of endairA (2)

sketch of the proof (cont' d):

- in the reduced configuration adversarial nodes are non-adjacent
- thus each sequence of non-repeating IDs has a unique partitioning
  - IDs of honest nodes form distinct partitions
  - consecutive adversarial IDs form a partition
- if the route is non-plausible, then (at least) one of the following must hold:
  - P_j={N_i} and P_{j+1}={N_{i+1}} are non-adversarial partitions and the nodes v and v' that belong to N_i and N_{i+1} are not adjacent in $\underline{G}$
  - P_j={N_i}, P_{j+1}={$N_{i+1}, \dots, N_{i+k}$}, P_{j+2}={N_{i+k+1}} are two non-adversarial (P_j, P_{j+2}) and an adversarial partition (P_{j+1}) and the nodes that belong to N_j and N_{j+k+1} have no common neighbor that belongs to V*
- in the first case, N_i would detect that the next ID in the list doesn' t belong to a neighbor and wouldn' t sign the message
- in the second case, the route reply message cannot reach N_i
- note also that N_i sees the same list as S because it verifies the signature of D

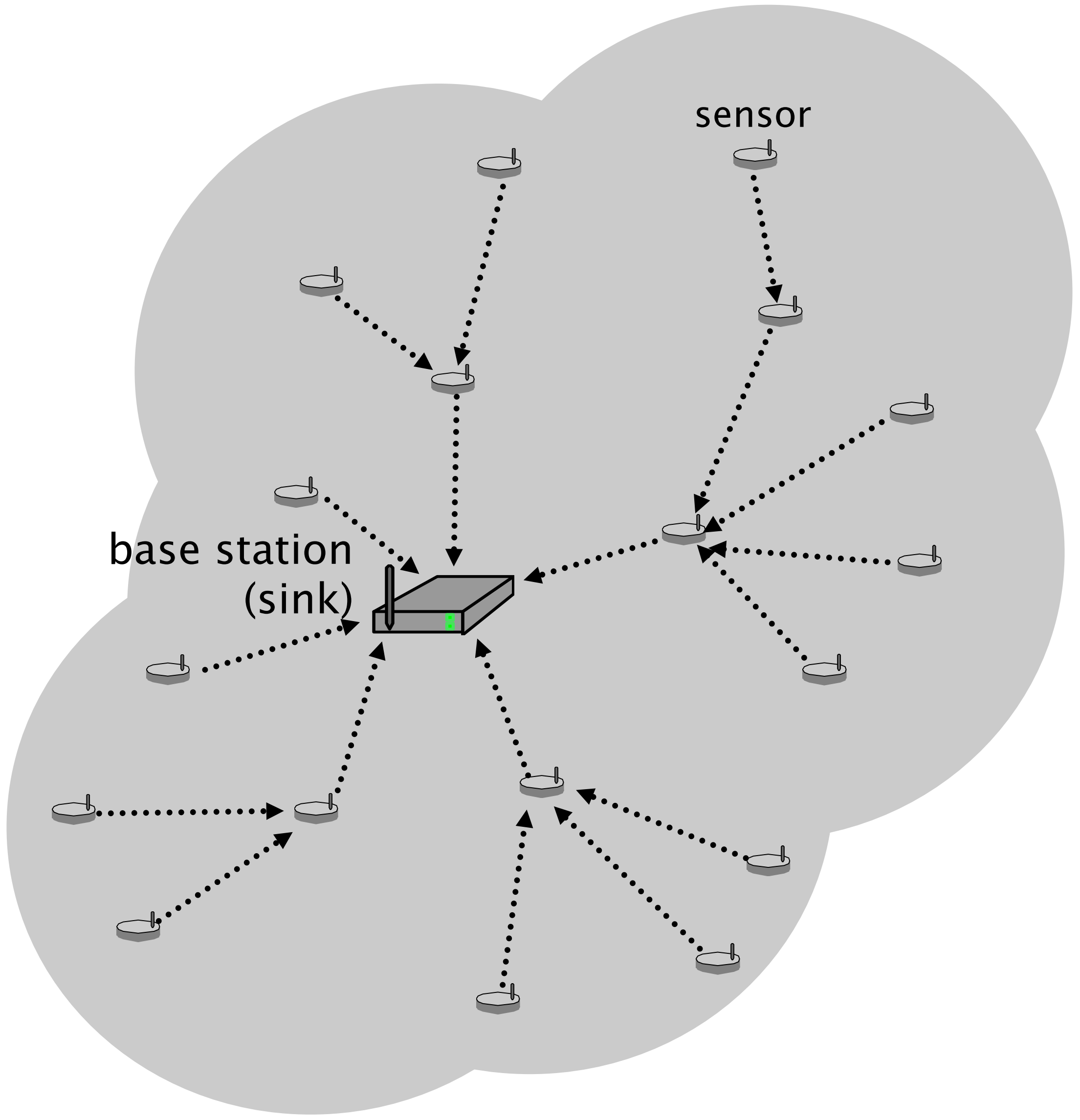
→ the adversary must have forged some signatures

Chapter Outline

How are sensor networks different?

- communication patterns
  - sensors to base station (many-to-one)
  - base station to sensors (one-to-many)
- limited mobility
  - sensor nodes are mainly static
  - topology can change due to node and link failures
  - much less dynamicity than in ad hoc networks of mobile computers
- resource constraints
  - sensor nodes are much more constrained in terms of resources
- infrastructure support
  - the base station can act as a trusted entity

TinyOS beaconing



Authenticated TinyOS beaconing

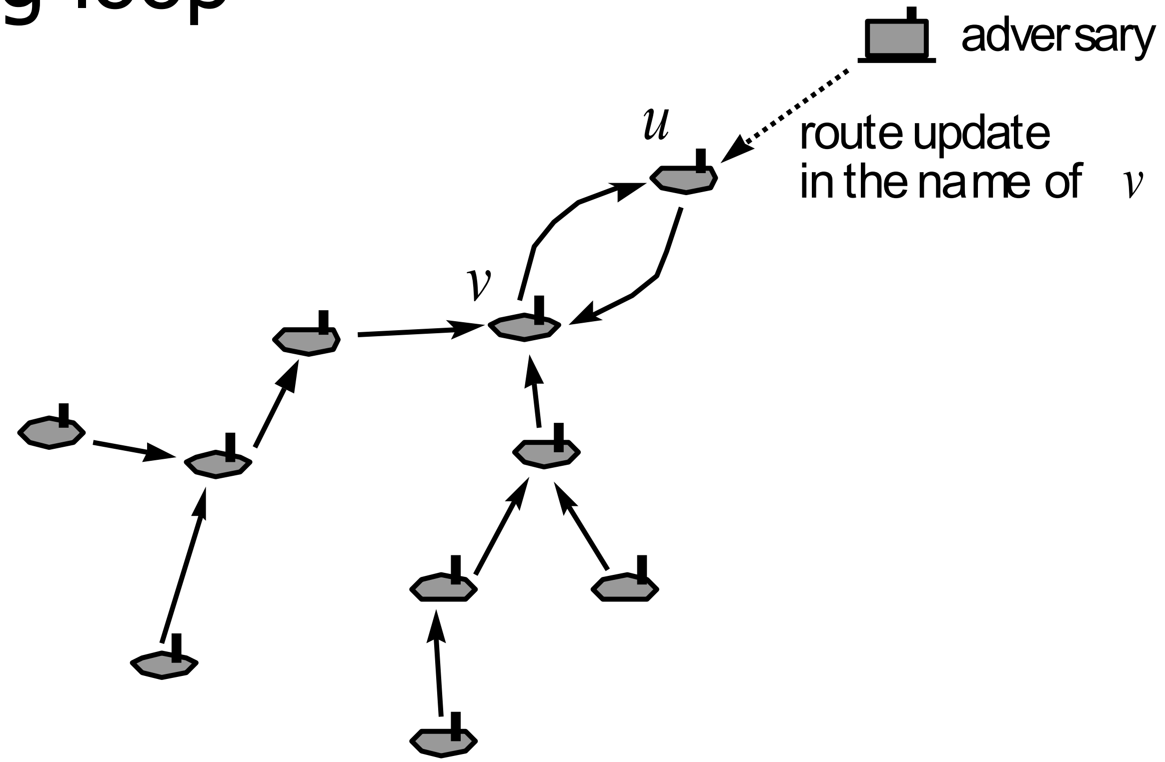
- Since beacon messages are not authenticated, an adversary can initiate the route update process and become the root of the established tree

- In order to prevent this, the base station should authenticate the beacon
  - needs broadcast authentication
  - due to resource constraints, symmetric key crypto should be used
  - a possible solution is TESLA

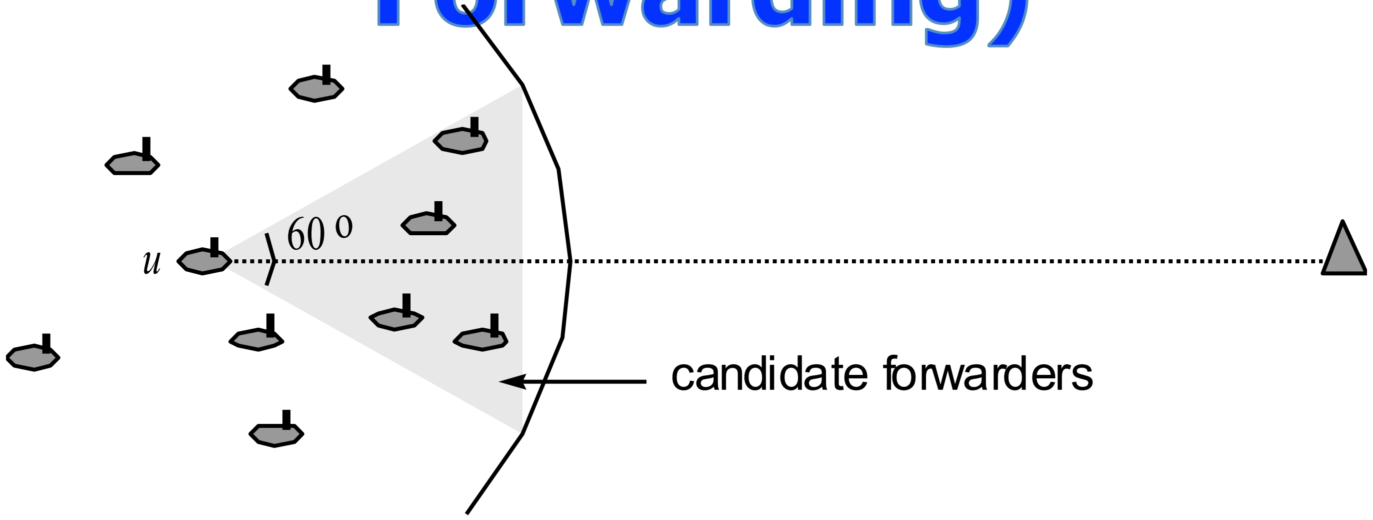
- This does not entirely solve the problem …

Authenticated TinyOS beaconing

- intermediate nodes are not authenticated
- an adversary can use spoofing to create a routing loop



IGF (Implicit Geographic Forwarding)



- position-based routing integrated with the RTS/CTS handshake of the MAC layer
- when u wants to send a packet, it broadcasts an RTS
  - contains the position of u and that of the destination
- neighbors in the 60^o sextant set their CTS timer inversely proportional to the weighted sum to their distance from u, remaining energy, and distance to the line between u and the destination
  - most desirable next hop will send CTS first
- all other nodes hear the first CTS and cancel their timers

Securing IGF

- an adversarial node can send CTS immediately and become the next hop
  - nodes should not cancel their CTS timers
  - u waits until more neighbors send CTS, and selects the next hop randomly
- an adversary can masquerade as many different potential next hop neighbors and increase her chances to be selected as the next hop
  - neighbors should be authenticated and next hop should be selected from the set of authenticated neighbors
- an insider adversary can still use her compromised identifiers
  - monitoring the behavior of neighbors (???)
  - those that often fail to forward packets should not be selected as next hop

Summary

- routing is a fundamental function in networking, hence, an ideal target for attacks
- attacks against routing aim at
  - increasing adversarial control over the communications between some nodes;
  - degrading the quality of the service provided by the network;
  - increasing the resource consumption of some nodes (e.g., CPU, memory, or energy)
- many attacks (but not all!) can be prevented by authenticating routing control messages
- it is difficult to protect the mutable parts of control messages
- special attacks (e.g., tunnels and rushing) needs special protection mechanisms
- several secured ad hoc network routing protocols have been proposed
- some of them have weaknesses that are exploitable by attacks