Oral Exams

-Will take place **starting from 11.12.2017** (next week)

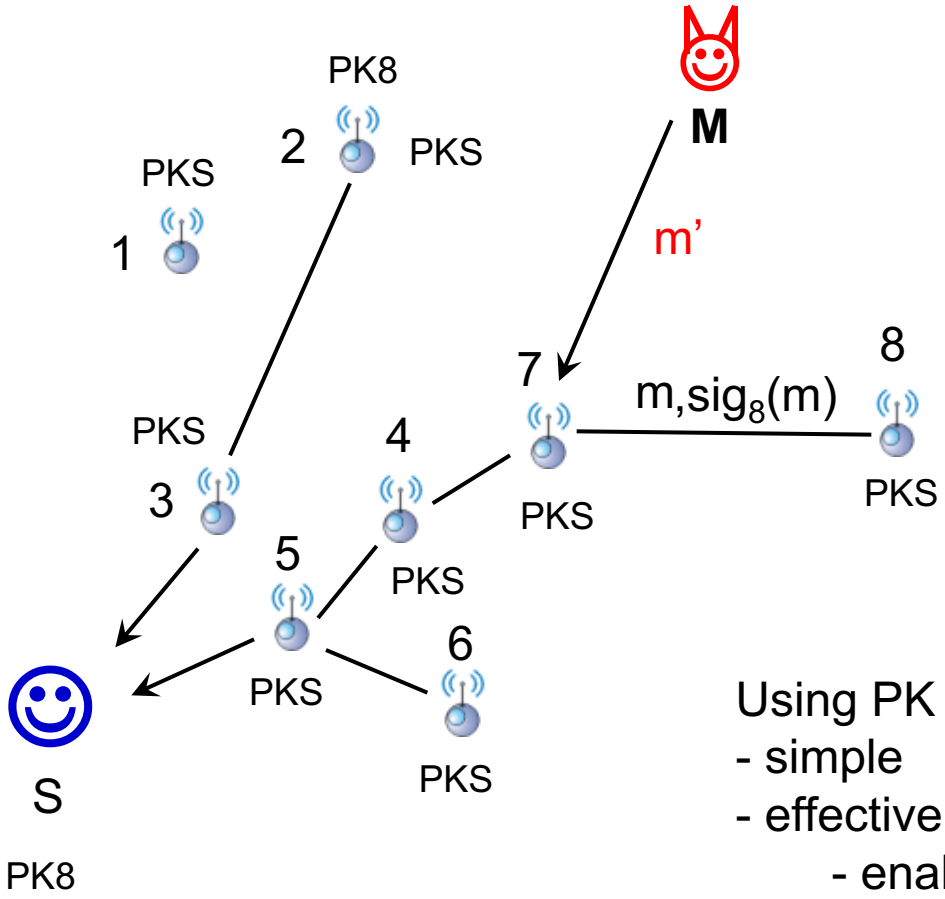
-You should have received already a doodle link

-If not, email **sown@lists.inf.ethz.ch** and get this sorted out

-20 min roughly per student

Key Distribution in Sensor Networks

Data integrity, authentication



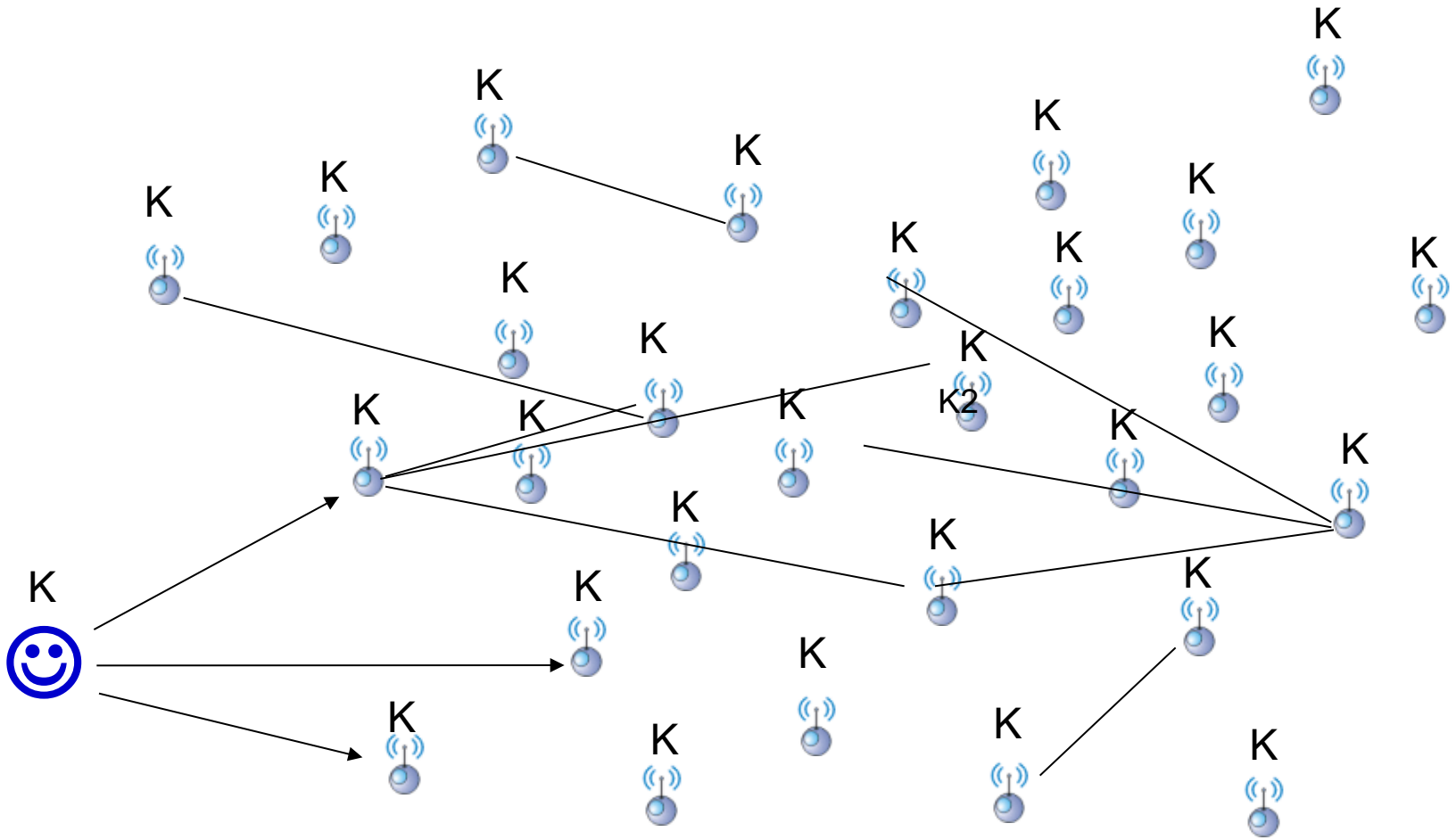
Using PK crypto in distributed networks is:
- simple
- effective
  - enables broadcast authentication
  - distribution of new keys and insertion of new nodes is straightforward

- expensive

Symmetric-key and PK crypto in sensor nets

- Use PK for all operations
  - + simple key distribution
  - + simple broadcast authentication
  - – sensors need to be able to perform PK crypto
- PK for key establishment (DH) and SK for the rest
  - + simple key distribution
  - – no efficient broadcast authentication
  - – sensors need to be able to perform SK and PK crypto
- Use SK for all operations
  - – **key distribution becomes an issue**
  - – **no efficient broadcast authentication**
  - + sensors need to be able to perform only SK crypto

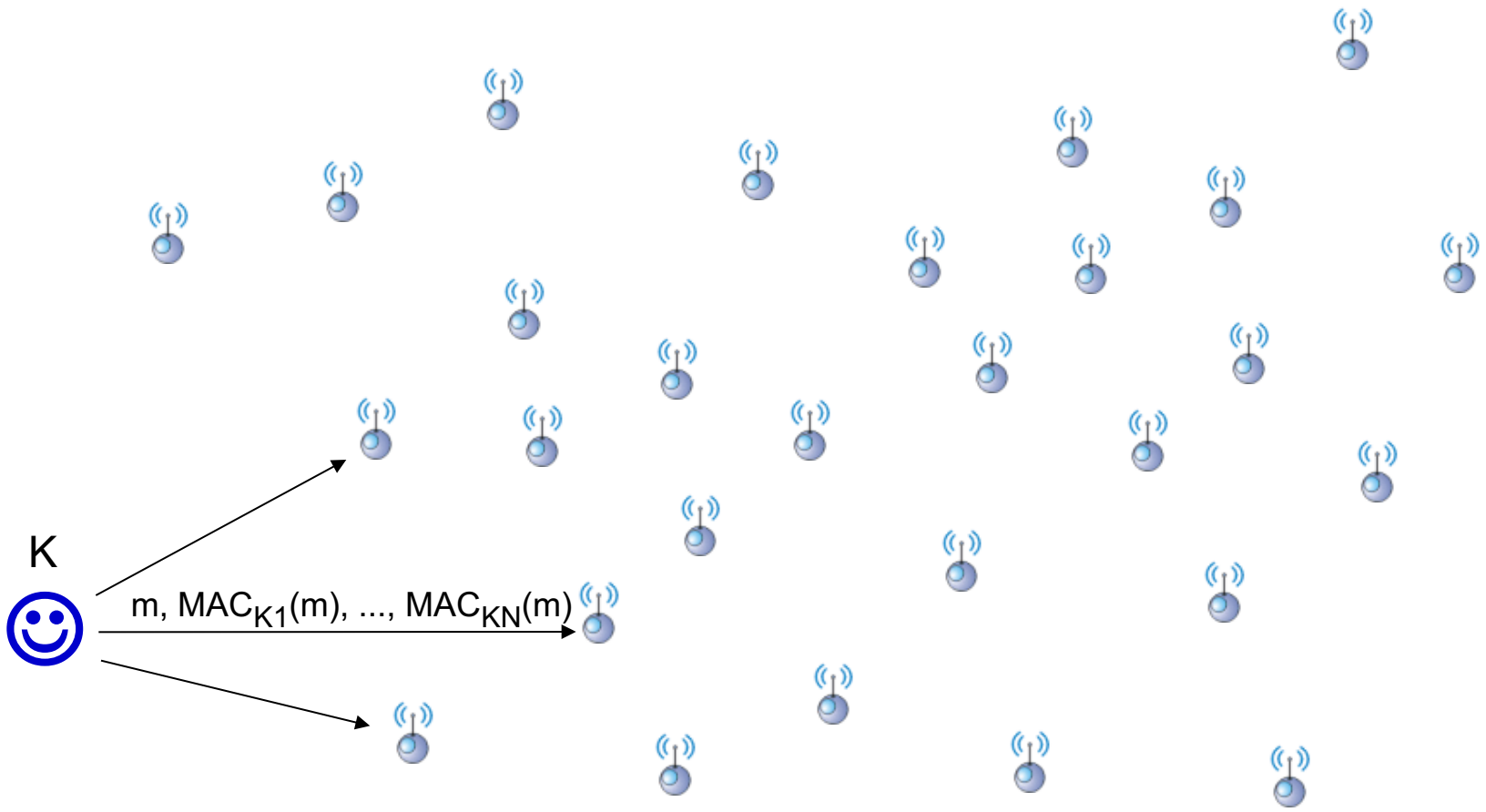
(S)Key distribution in sensor networks [Eschenauer, Gligor]



1 key for all network nodes

+ low storage (1key)
+ efficient broadcast authentication
- no resilience to compromise
- easy to add new nodes

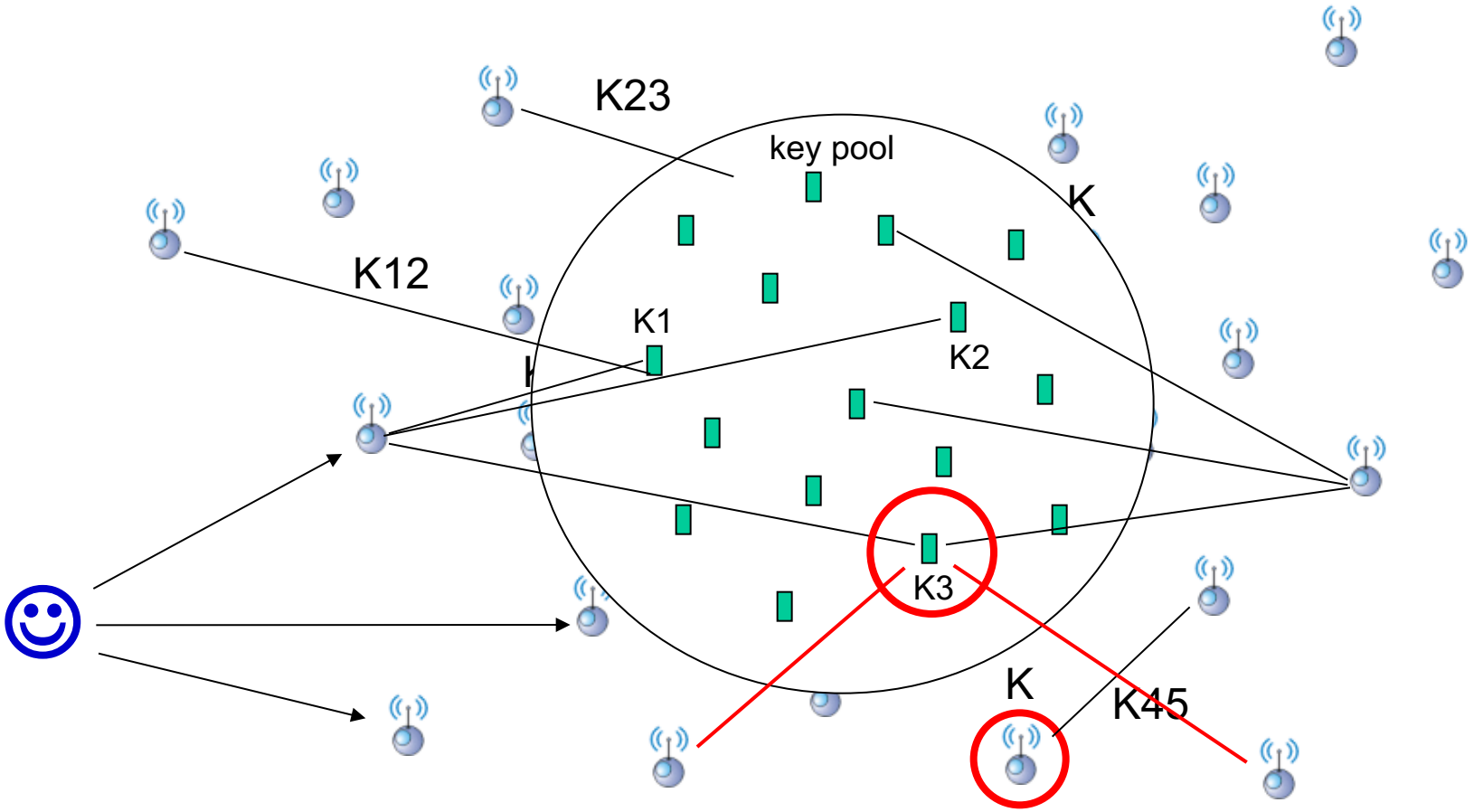
(S)Key distribution in sensor networks [Eschenauer, Gligor]



Each node pair has a different key

- high storage (n keys)
- inefficient broadcast authentication
+ resilience to node compromise
- expensive to add new nodes

(S)Key distribution in sensor networks [Eschenauer, Gligor]



Some node pairs end-up with the same keys

- lower storage (sqrt(n) keys)
- inefficient broadcast authentication
+ some resilience to node compromise
+ easy to add new nodes

(S)Key distribution in sensor networks

Main idea:

- instead of preloading *n* keys in each node, preload just a small subset of values (k<<n) that make sure that most nodes (probabilistic) or all nodes (deterministic) establish keys

Placed between two extremes:

- single master key *(1)*
- distinct pair-wise keys for all node pairs *(n^2)*

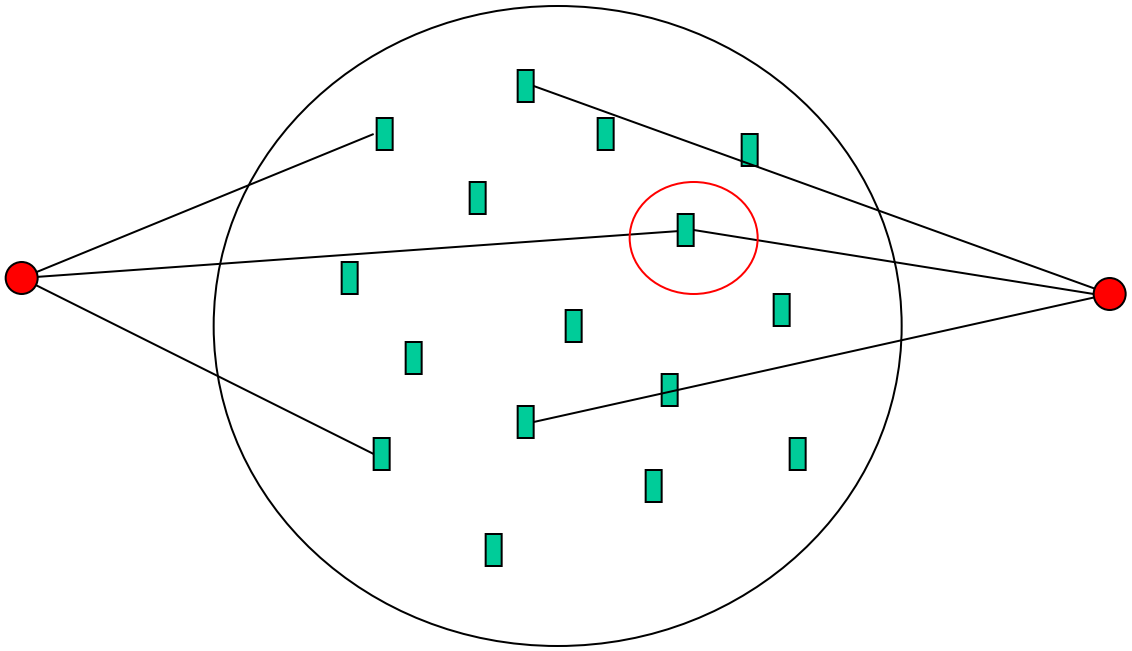
Main issues

- Computation (per key established)
- Communication (per key established)
- Memory (sensor storage)
- Key sharing graph connectivity
- Resiliency (how many sensors need to be compromised before the entire pool is disclosed)
- Scalability

[EG] Scheme

Basic probabilistic key pre-distribution

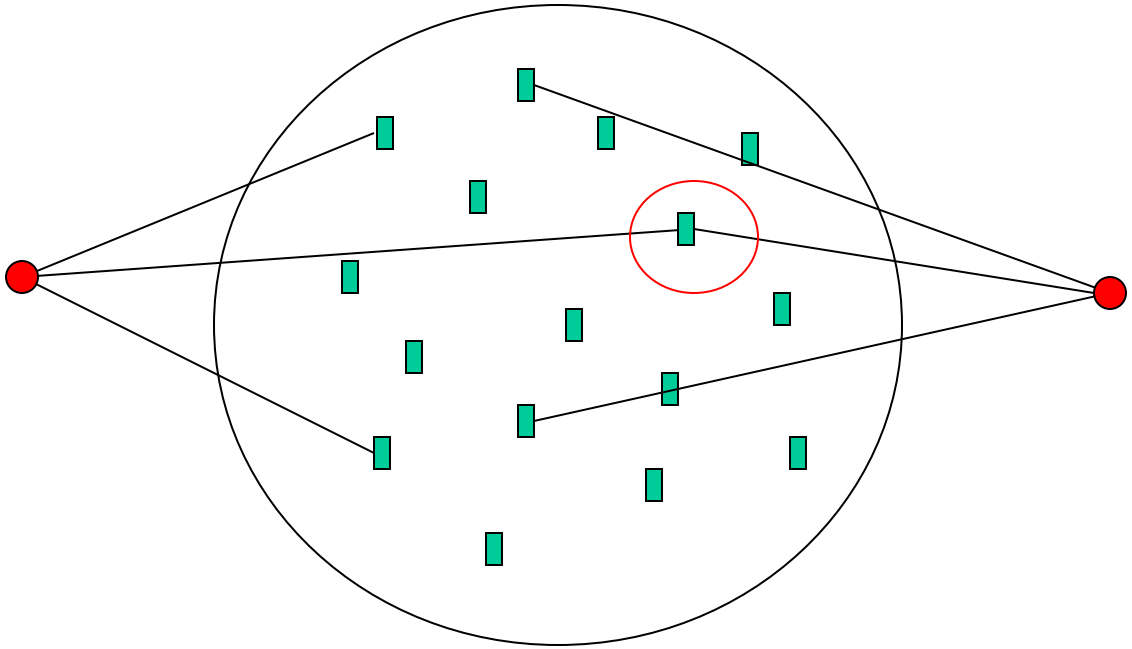
- Eschenauer and Gligor (EG), CCS 2002



k keys in the pool ; sqrt(k) stored per node

[EG] Scheme

- **Key setup prior to deployment:**
  keys are generated and loaded into memory (the whole pool is known only to the authority)
- **Shared-key discovery after deployment:**
  each sensor node broadcasts a key identifier list to *one-hop neighborhood* (more than one pair may share the same key)
- **Path-key establishment:**
  if two sensor nodes still do not share a key



[EG] Probability of sharing a key

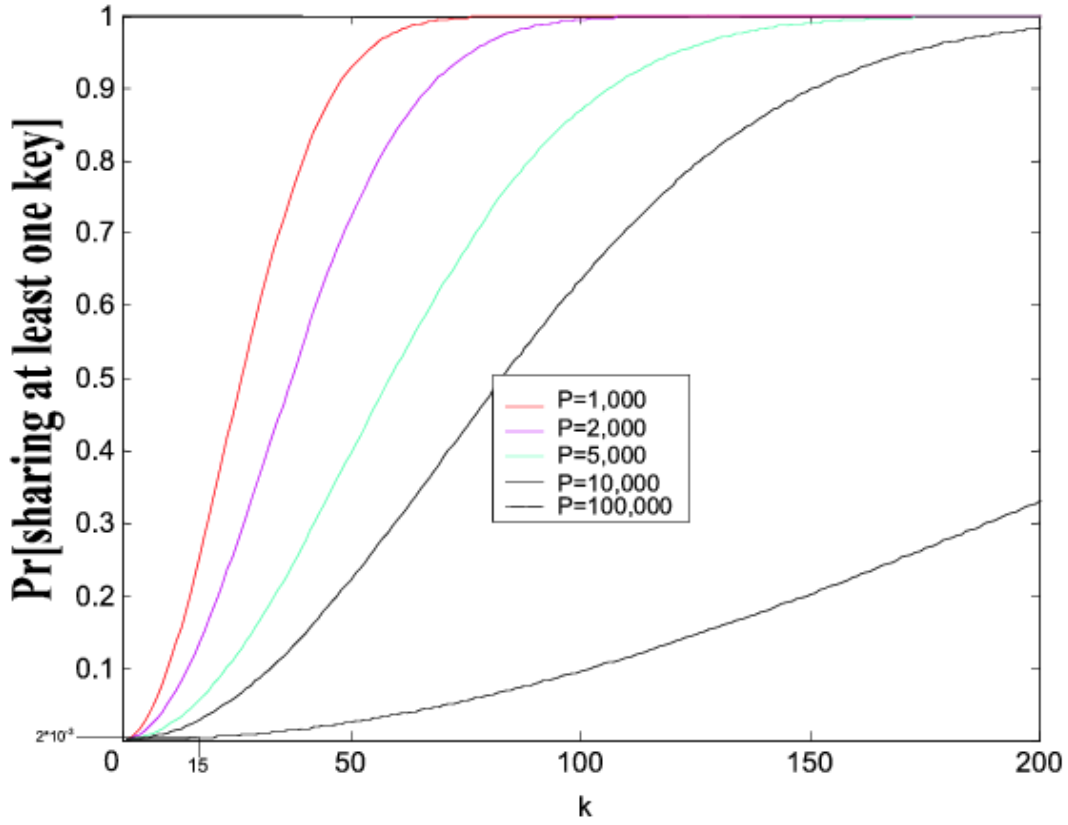
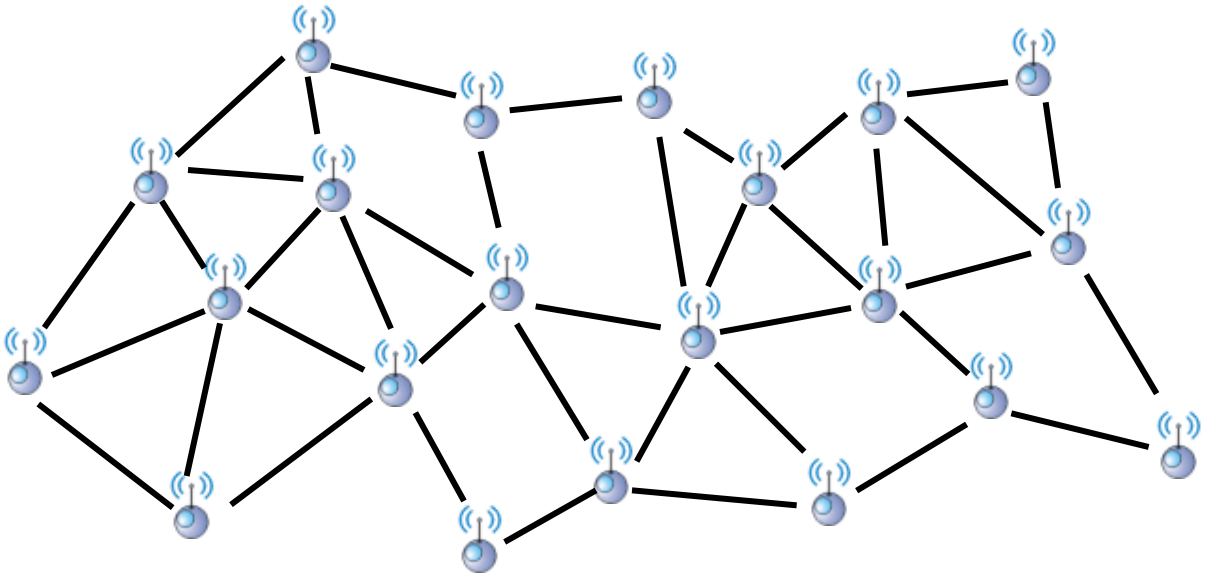


Figure 2: Probability of sharing at least one key when two nodes choose k keys from a pool of size P

[EG] Key Graph and Key Sharing Graph

- Key graph $G_k(V,E)$ is defined as follows:
  - V represents all the nodes in the sensor net
  - For any tow nodes i and j in V, there exists an edge between them if and only if :
    - 1) i and j share at least one common key
- Key sharing graph $G_{sk}(V,E')$
  - i and j have an edge if and only if
    - 1) And 2) They are within wireless transmission range



[EG] Key Graph and Key Sharing Graph

- Key graph $G_k(V,E)$ is defined as follows:
  - V represents all the nodes in the sensor net
  - For any tow nodes i and j in V, there exists an edge between them if and only if :
    - 1) i and j share at least one common key
- Key sharing graph $G_{sk}(V,E')$
  - i and j have an edge if and only if
    - 1) And 2) They are within wireless transmission range

Better connected Key sharing graph = increased communication ability/security

Better connected key graph = increased vulnerability to compromise **...**

[EG] Connectivity vs. Resiliency

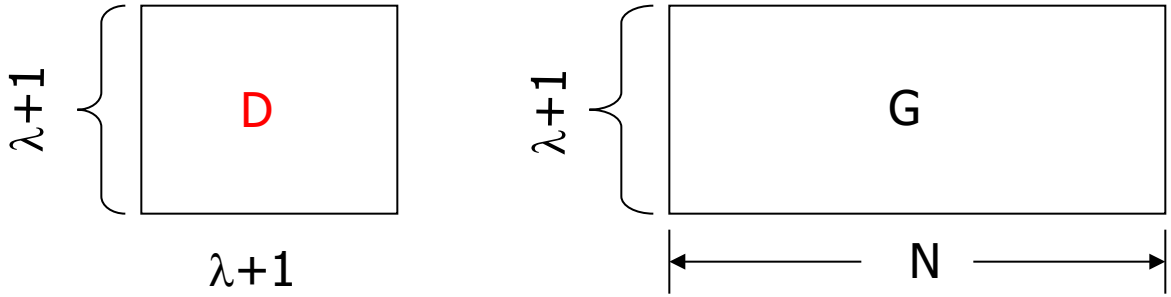
- The contradictory requirement on Key Pool size |P|
  - Larger key pool size – better resiliency
  - Smaller key pool size – better connectivity
- The key pool size is restricted by network size
  - **$|P| < k^2/\ln(1/(1-p))$**
    p is the probability that two nodes share a key (k – number of stored keys)
  - **$p > O(\ln N)/n$**
    N is the number of sensor nodes in the network and n is the average node degree.
  - As N increases, in order to maintain connectivity, p would increase, which leads to shrink in $|P|$
- Property of resiliency does not scale with network size
  - p should be non decreasing as network enlarges.
  - compromising k nodes compromises kp links

Deterministic Approaches

- Used to design the key pool and the key chains to provide better connectivity
  - Matrix Based Scheme [Blom 1985]
  - Polynomial Based Key Generation [Blundo et al. 1992]

Deterministic approaches: Blom's Scheme [B]

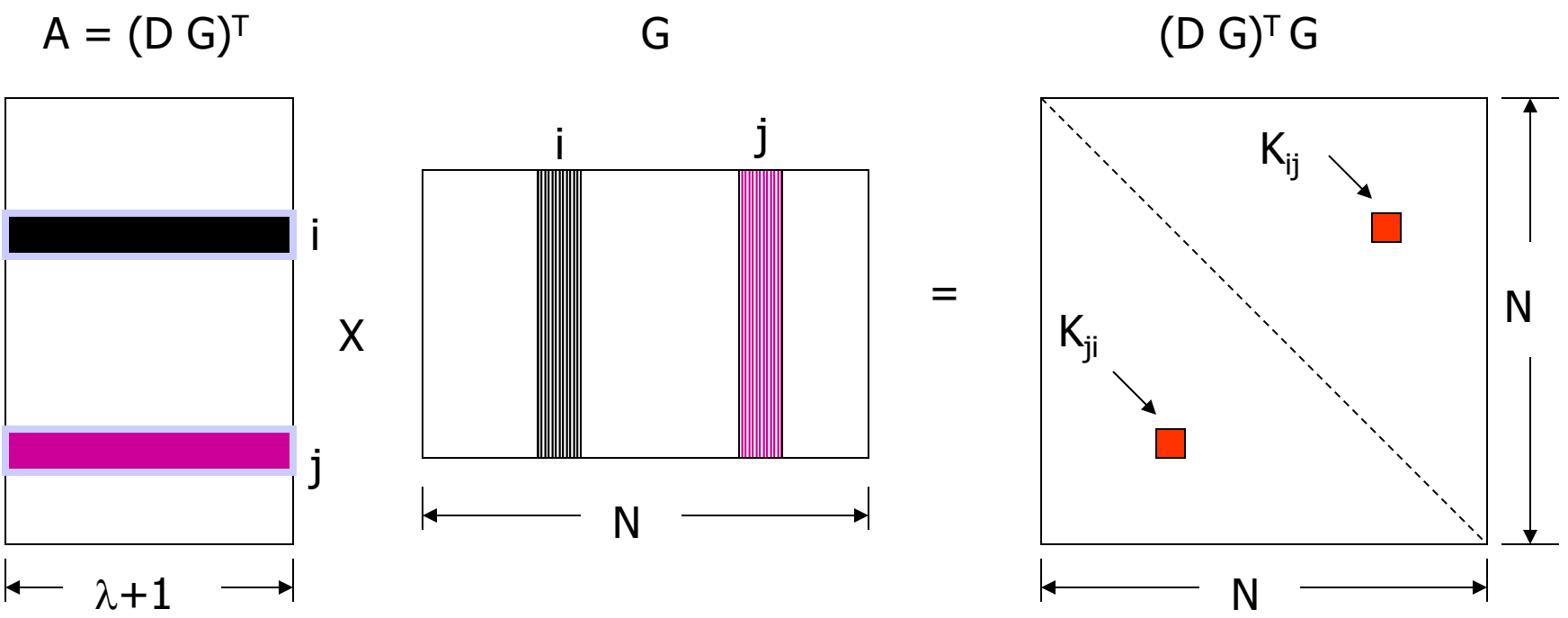
- Public matrix G
- Private matrix D (symmetric).



Let $A = (D\ G)^T$

$$A\ G = (D\ G)^T\ G = G^T\ D^T\ G = G^T\ D\ G = (A\ G)^T$$

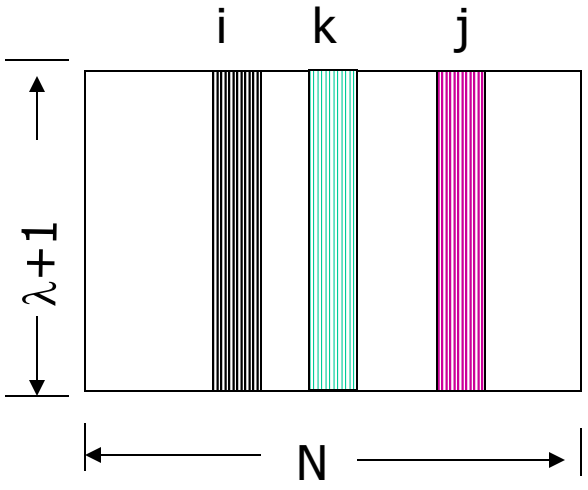
[B] Scheme



Node i carries:

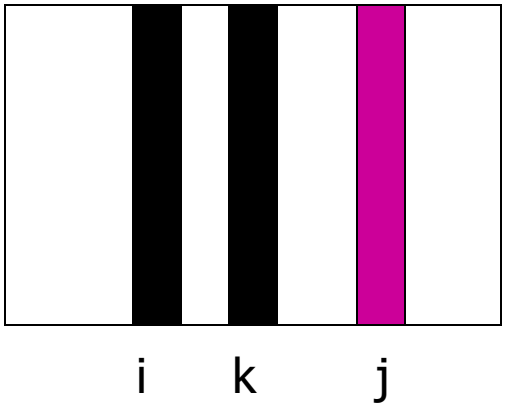
Node j carries:

[B] λ-secure Property



Undesirable Situation:
if
$u*G(i) + v*G(j) = G(k)$

then
$u*A(i) + v*A(j) = A(k)$



$A^T=D\ G$

this would allow
colluding nodes (i and j) to
impersonate other nodes (k)

[B] λ-secure Property

- **ALL** **λ+1** columns in G are linear independent.
  - Different from saying that G has rank **λ+1**
  - **Rank:** there are **λ+1** lineary independent columns
- Can tolerate compromise up to λ nodes.
  - Once **λ+1** nodes are compromised, the rest can be calculated if these **λ+1** columns are linear independent.
- How to find such a matrix G?

[B] Vandermonde Matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ s & s^2 & s^3 & \cdots & s^N \\ s^2 & (s^2)^2 & (s^3)^2 & \cdots & (s^N)^2 \\ & & \vdots & & \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \cdots & (s^N)^\lambda \end{pmatrix}$$

[B] Properties of Blom Scheme

- Blom's Scheme
  - Network size is N
  - Any pair of nodes can directly find a secret key
  - Tolerate compromise up to λ nodes
  - Need to store λ+2 keys

Key distribution schemes for sensor networks

http://www.cs.rpi.edu/research/pdf/05-07.pdf

Problem	Approach	Mechanism	Keying style	Papers
Pair-wise	Probabilistic	Pre-distribution	Random key-chain	C, E, F, J K, N, S
			Pair-wise key	E
	Deterministic	Pre-distribution	Pair-wise key	G, M
			Combinatorial	P, Q
		Dynamic Key Generation	Master key	D, L
			Key matrix	A
			Polynomial	B, G
	Hybrid	Pre-distribution	Combinatorial	P, Q
		Dynamic Key Generation	Key matrix	H, M, R
			Polynomial	I, R
Group-wise	Deterministic	Dyn. Key Gen.	Polynomial	B, R

The papers are: A[Blom 1985], B[Blundo et al. 1992], C[Eschenauer and Gligor 2002], D[Lai et al. 2002], E[Chan et al. 2003], F[Pietro et al. 2003], G[Liu and Ning 2003c], H[Du et al. 2003], I[Liu and Ning 2003b], J[Zhu et al. 2003], K[Du et al. 2004], L[Dutertre et al. 2004], M[Lee and Stinson 2004b], N[Hwang et al. 2004], P[Camtepe and Yener 2004], Q[Lee and Stinson 2004a], R[Huang et al. 2004], S[Hwang and Kim 2004].

Confidentiality and Integrity, Authentication for Sensor Networks using SK primitives

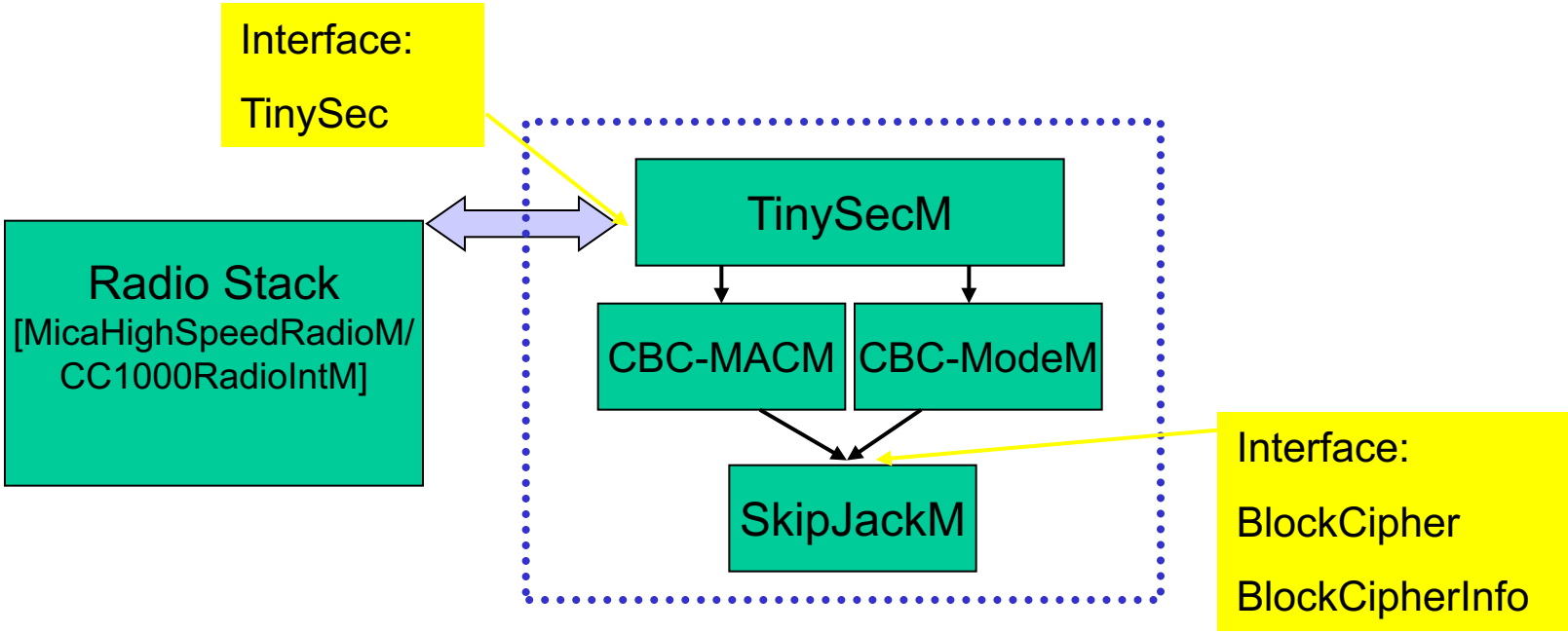
[TinySec]: Encryption and MAC for Sensor Networks

- A Link Layer Security Architecture for Wireless Sensor Networks, Chris Karlof, Naveen Sastry, and David Wagner.
  ACM SenSys 2004,

- Implementation of **encryption and MAC** on Mica motes running TinyOS.
  - main challenges in code size/speed of execution/memory

- Uses **Skipjack block cipher** for both encryption and MAC

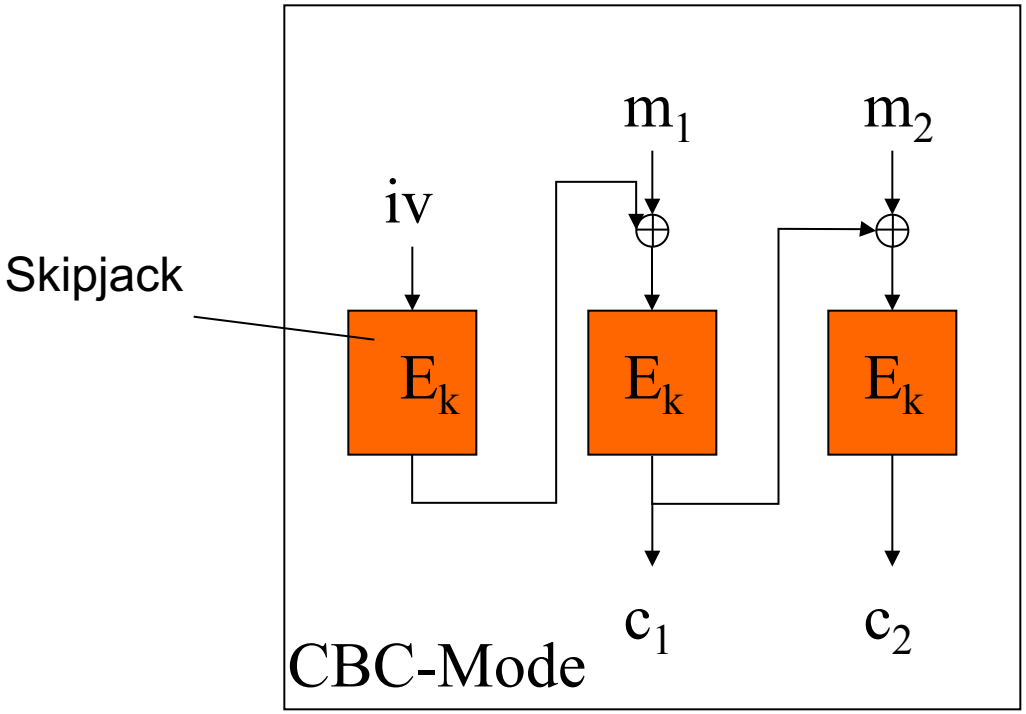
[TinySec] Components



- Use a block cipher for both encryption & authentication
- Skipjack is good for 8-bit devices; low RAM overhead

[TinySec] Confidentiality

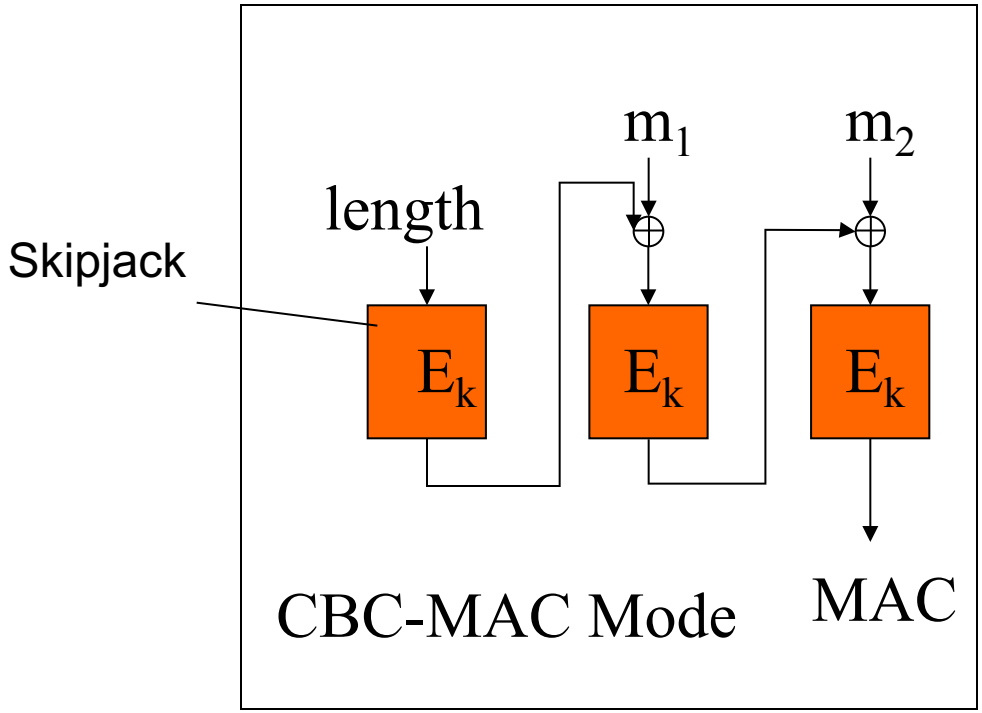
- Confidentiality achieved by encryption
  - CBC (cipher-block chaining)-mode: break a *m* bit message into 64 bit chunks (m1,m2,..) Transmit (c1, c2, …)
  - **iv** is needed to achieve *semantic security (*A message looks different every time it is encrypted).
  - **iv** reuse is "prevented"



[TinySec] Integrity, Authentication

- Integrity achieved by a message authentication code
  - A *t* bit cryptographic checksum with a *k* bit key from an *m* bit message

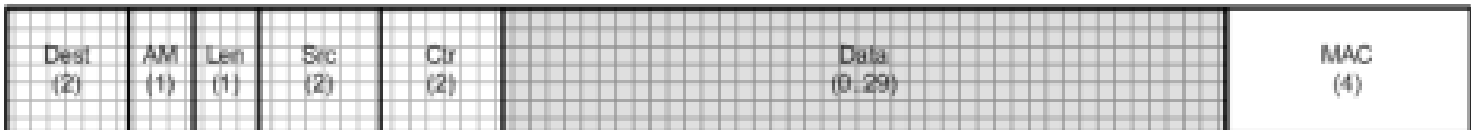
  Can detect both malicious changes and random errors
  - Replaces CRC can be built using a block cipher
  - MAC key different than encryption key



[TinySec] Packet formatting: low comm. overhead

Old packet (CRC): +7 b

Authentication Only (TinySec-Auth): +8 b



IV

Authentication, Encryption (TinySec-AE) : +12 b

	Overhead (b)	Total Size (b)	Xmit time (ms)	Increase
CRC	39	63	26.2	--
TinySec-Auth	40	64	26.6	1.5%
TinySec-AE	44	68	28.8	8%

[TinySec] Implementation on Mica motes

- Implementation
  - 3000 lines of NesC code
  - RAM: 455 bytes (not an issue for applications, can be reduced to 256 bytes)
  - MEM: 7000 bytes of program space
  - **Real time: Two priority TinyOS scheduling process (cryptographic computations must be completed by the time the radio finishes sending the start symbol)**

[SNEP]: Sensor-Network Encryption Protocol

[SNEP] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar, Mobicom 01

- A and B share
  - Encryption keys: K_{AB} K_{BA}
  - MAC keys: K'_{AB} K'_{BA}
  - Counters: C_A C_B
- To send data m, A sends to B:

$A \rightarrow B$: $\{m\}_{<K_{AB}, C_A>}$

$MAC(K'_{AB} , [C_A \,||\, \{m\}_{<K_{AB}, C_A>}])$

[SNEP] properties

- Secrecy & confidentiality
  - Semantic security against chosen ciphertext attack
- Authentication
- Replay protection
- Strong freshness protocol in the paper
- Code size: 1.5 Kbytes

Secure Routing in Wireless Networks

Srdjan Čapkun
Department of Computer Science
ETH Zurich

Material From L. Buttyan and JP Hubaux (Security and Cooperation in Wireless Networks)

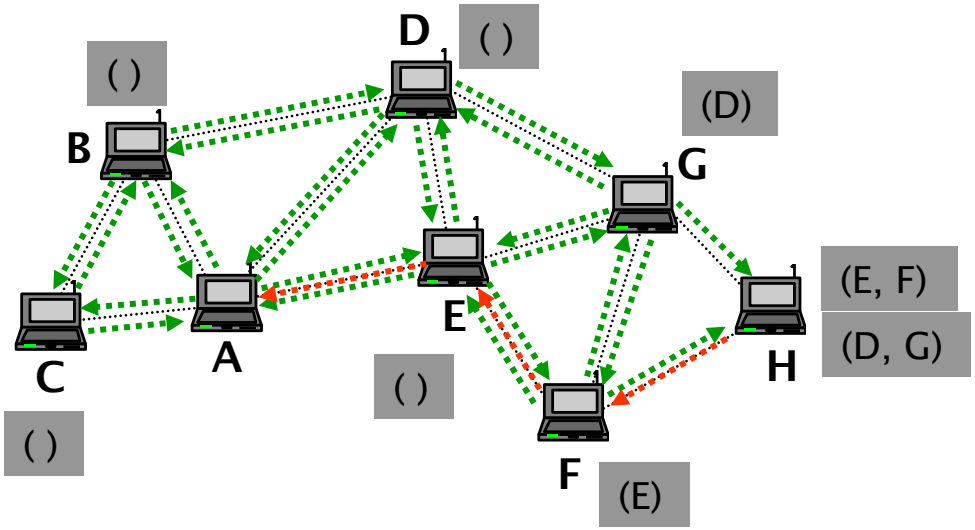
Ad hoc network routing protocols

- topology-based protocols
  - proactive
    - 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

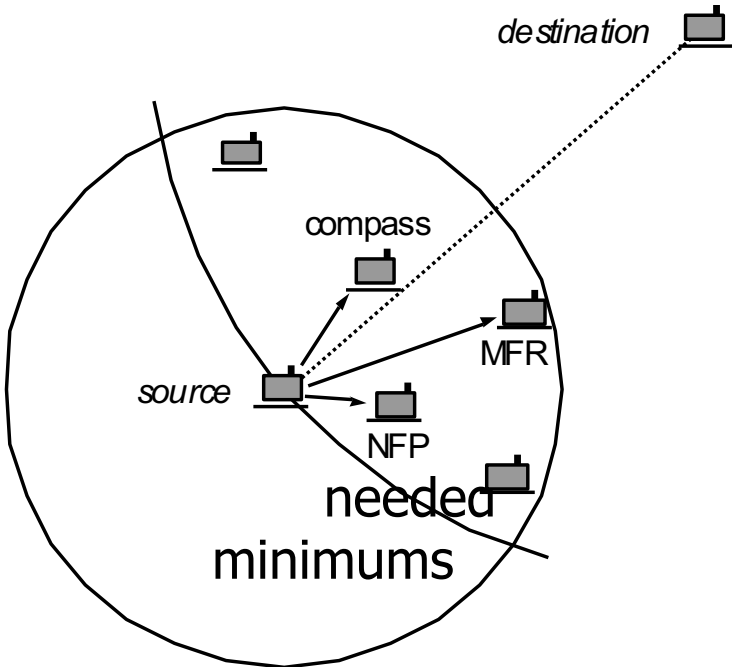
- from the route cache of H
- by reversing the route received in the RREQ
  - works only if all the links along the discovered route are bidirectional
  - IEEE 802.11 assumes that links are bidirectional
- by executing a route discovery from H to A
  - discovered route from A to H is piggy backed to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

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)



Secure Routing: Attacks

Attacks on routing protocols (1/2)

- general objectives of attacks
  - increase adversarial control over the communications between some nodes;
  - degrade the quality of the service provided by the network;
  - 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
  - rushing

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

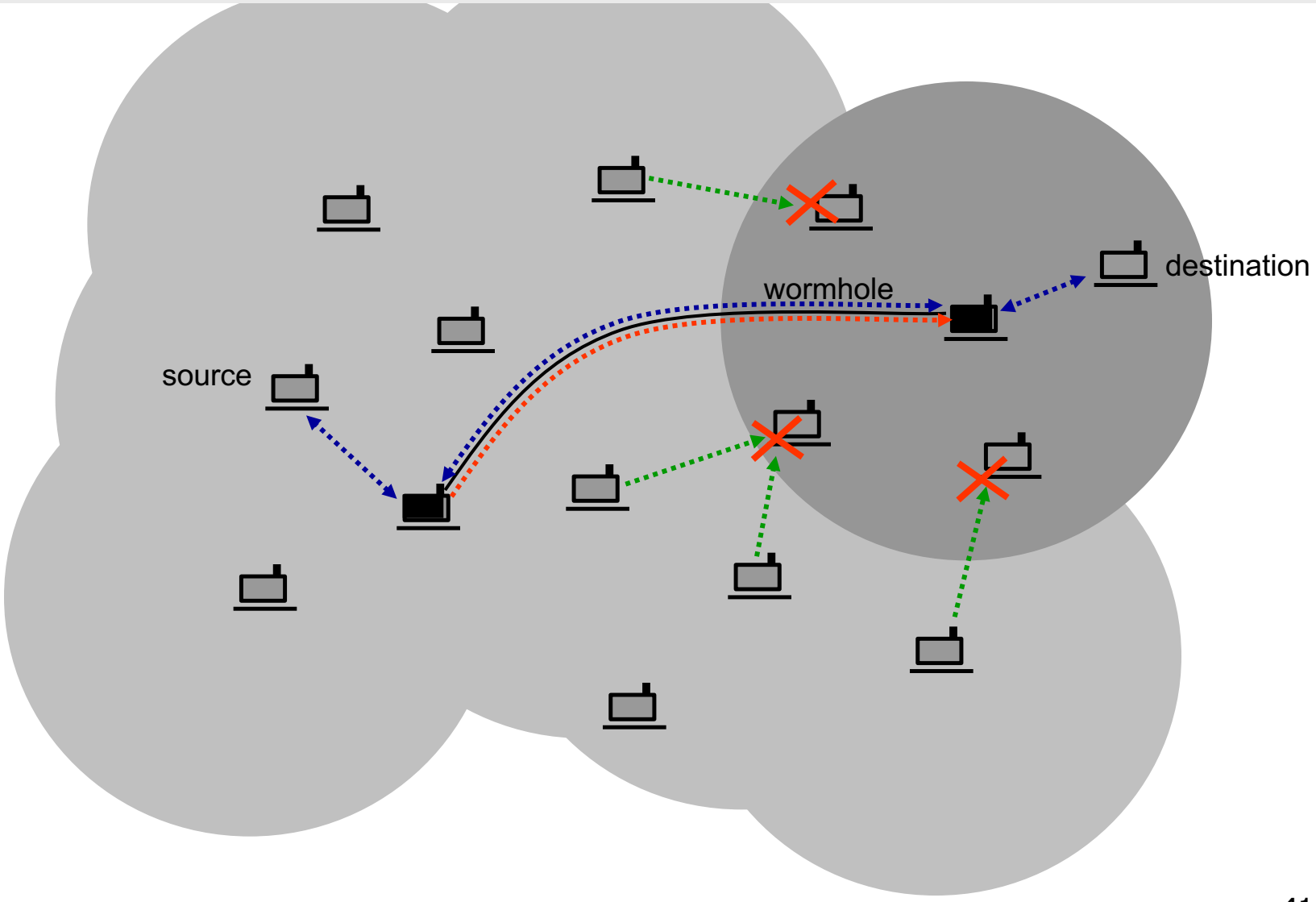
Route disruption

- 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

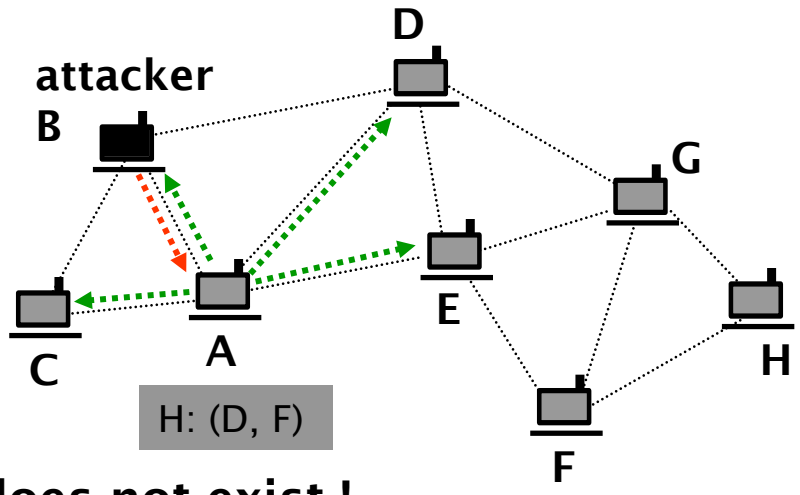
- the objective of route diversion can be
  - to increase adversarial control over the communications between some victim nodes
    - the adversary tries to achieve that the diverted routes contain one of the nodes that it controls or a link that it can observe
    - the adversary can eavesdrop or modify 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)]

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

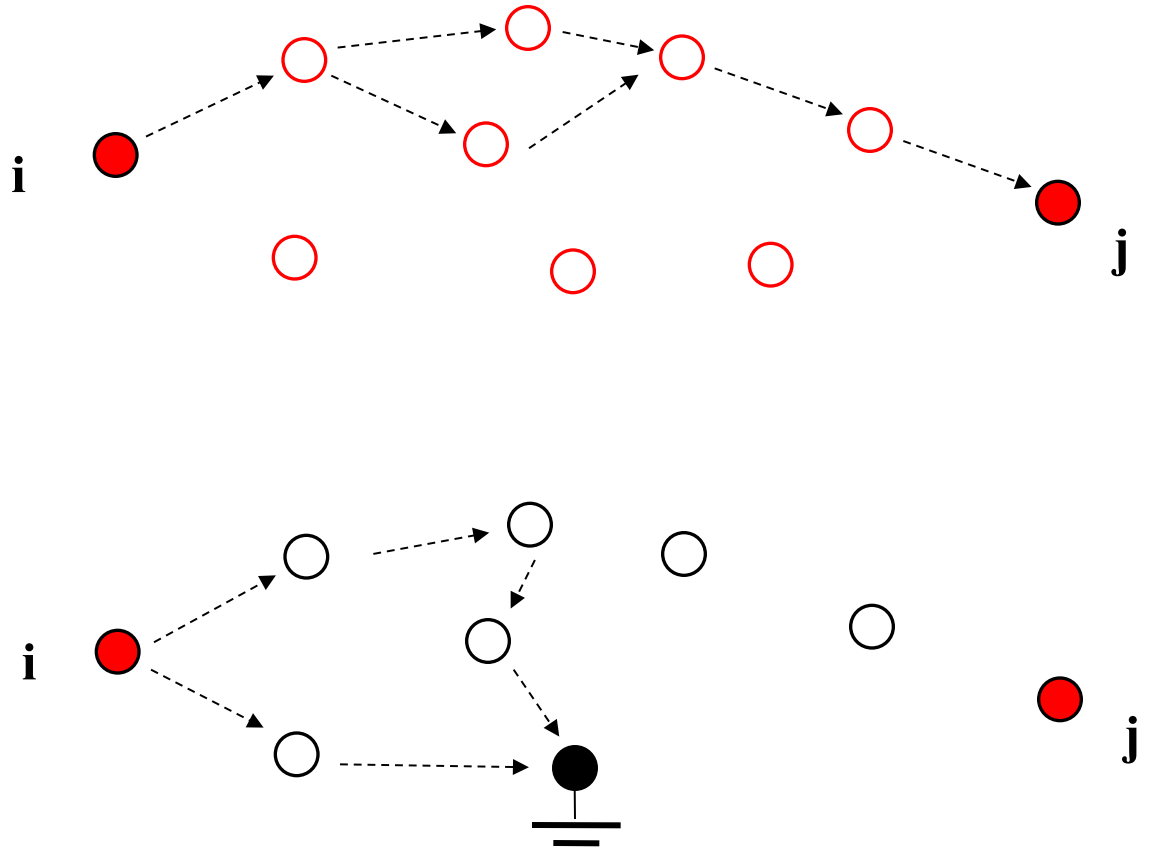
Setting up a gray/black hole

- 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

Black hole attack



Secure Routing:
Securing Routing Protocols

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

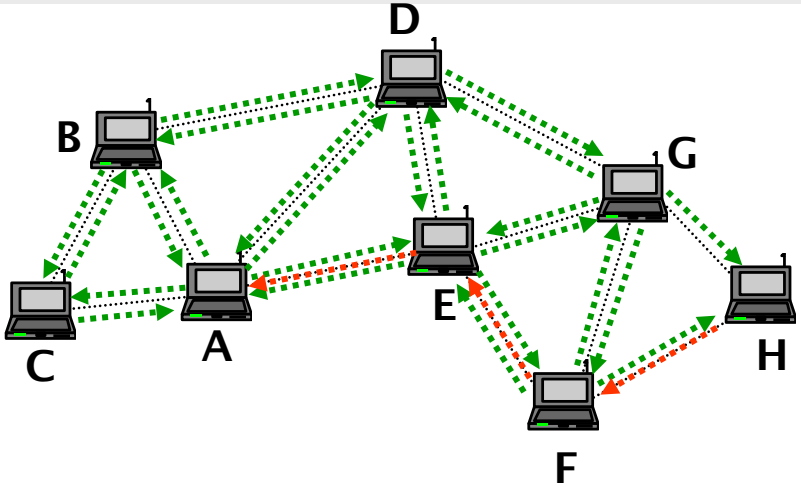
- uses symmetric-key authentication (MACs)
  - due to mobility, it would be impractical to require that the source and the destination share keys with all intermediate nodes
  - hence there's only a shared key between the source and the destination (?)

→ 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)]

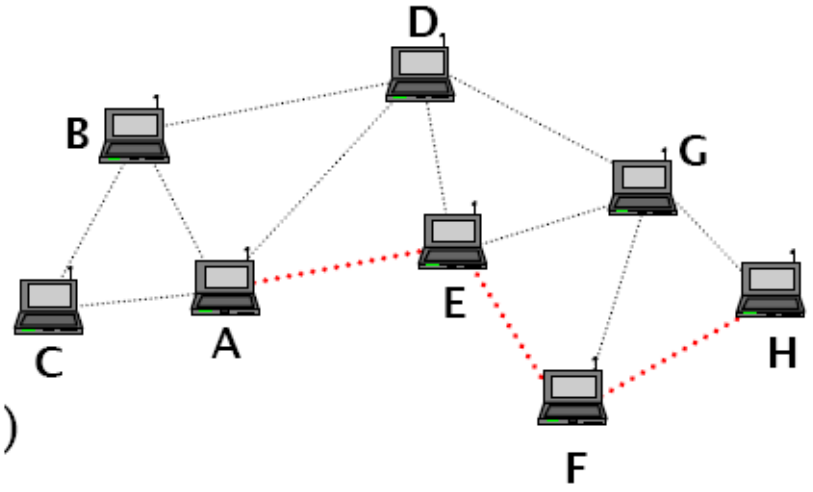
H → A : [RREP, A, H, id, sn, (E, F), mac_{HA}]

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 (mutable information)

Ariadne with signatures



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

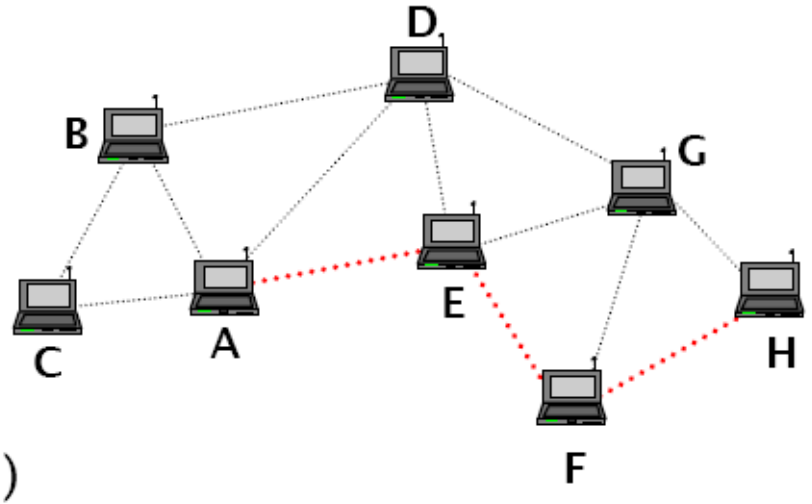
E : $h_E = H(E \mid h_A)$
E → * : [RREQ, A, H, id, h_E, (E), (sig_E)]

F : $h_F = H(F \mid h_E)$
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



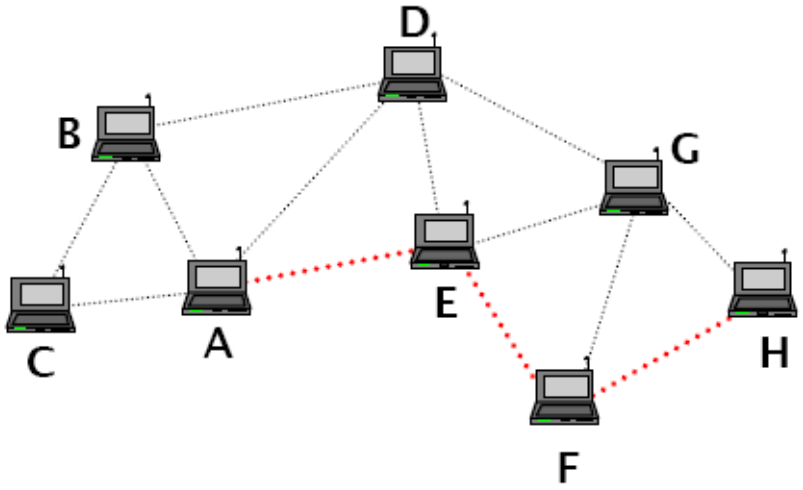
A : $h_A = mac_{AH}(RREQ \mid A \mid H \mid id)$
A → * : [RREQ, A, H, id, h_A, (), ()]

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

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

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

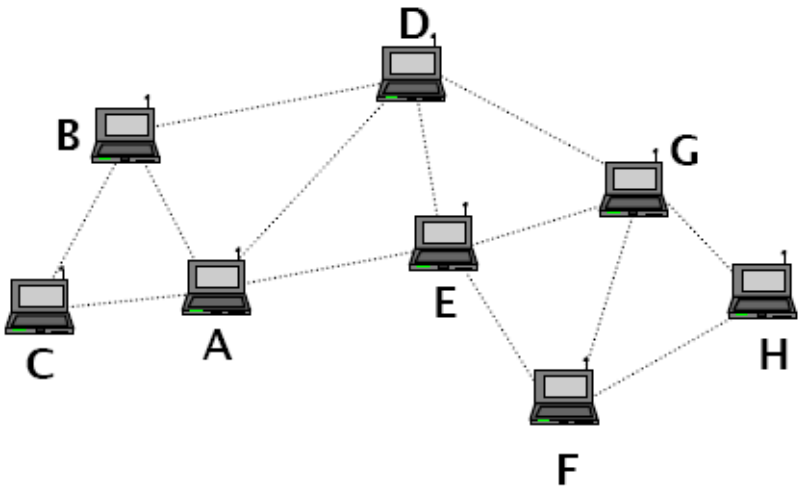
Ariadne with TESLA illustrated



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)]

Conclusion: Secure Routing Protocols

- Secure against Active-1-x attackers
- Efficient symmetric cryptography
- Generally better than DSR without optimization
- Source routing fits secure ad hoc network routing better than other routings
  - Sender can circumvent potentially malicious nodes
  - Sender can authenticate every node in Route Reply

- No built-in resiliency to wormhole attacks