Security and Privacy in Upcoming Wireless Networks

Key establishment in ad hoc networks

exploiting
- physical contact;
- mobility and vicinity;

Exploiting physical contact

- target scenarios
  - modern home with multiple remotely controlled devices
    - DVD, VHS, HiFi, doors, air condition, lights, alarm, …
  - modern hospital
    - mobile personal assistants and medical devices, such as thermometers, blood pressure meters, …
- common in these scenarios
  - transient associations between devices
  - physical contact is possible for initialization purposes
- the ***resurrecting duckling*** security policy
  - at the beginning, each device has an empty *soul*
  - each empty device accepts the first device to which it is physically connected as its master (imprinting)
  - during the physical contact, a device key is established
  - the master uses the device key to execute commands on the device, including the *suicide* command
  - after suicide, the device returns to its empty state and it is ready to be imprinted again

Does mobility increase or reduce security ?

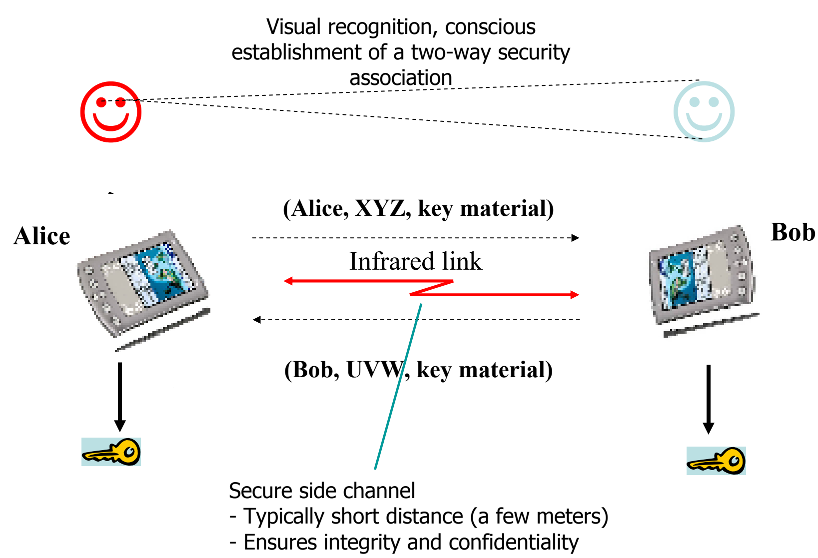
- Mobility is usually perceived as a major security challenge in networking
  - Wireless communications
  - Sporadic availability of the user/node
  - Higher vulnerability of the device
  - Reduced computing capability of the devices

- However, in real life, people often move (and gather) to increase security
  - Face to face meetings
  - Transport of assets and documents
  - Authentication by physical presence

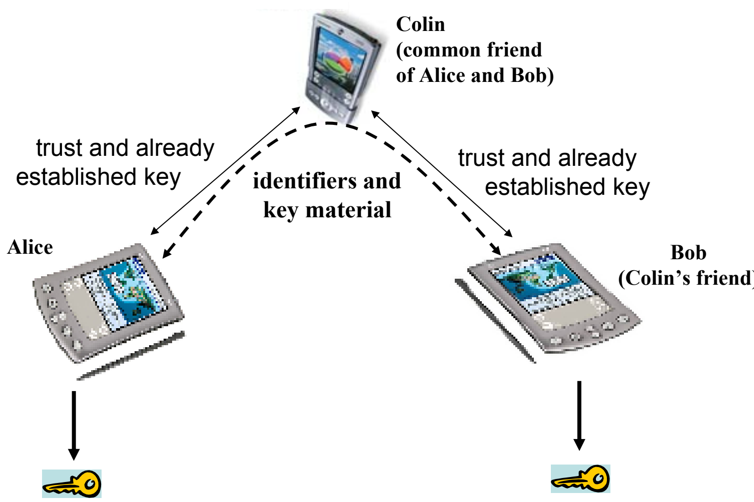
- Can we take advantage of mobility to increase security in networking?

- Yes, we can, assuming that
  - nodes are operated by humans
  - when in the vicinity of each other, nodes can use a ***secure side channel*** (e.g., infra red) to exchange a key
  - each node has some ***friends*** (peers that are trusted by the node), and there is already a key shared between each pair of friends

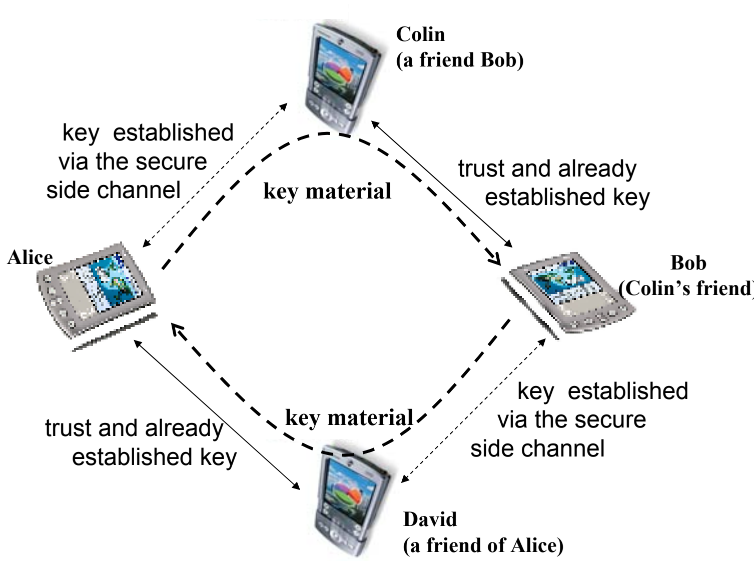
Exploiting vicinity and the secure side channel



Taking advantage of common friends



What if there's no common friend?



A possible implementation

$$\begin{aligned}
&\text{msg1} \; u \to v : f, r_u \\
&\text{msg2} \; v \to u : g, r_v \\
&\text{msg3} \; u \to g : u, \{d_{u\to g}, req, v, k_u, r_v\}k_{ug} \\
&\text{msg4} \; g \to v : g, \{d_{g\to v}, rep, u, k_u, r_v\}k_{vg} \\
&\text{msg3}' \; v \to f : v, \{d_{v\to f}, req, u, k_v, r_u\}k_{vf} \\
&\text{msg4}' \; f \to u : f, \{d_{f\to u}, rep, v, k_v, r_u\}k_{uf} \\
&\qquad u, v \; : k_{uv} = h(k_u, k_v)
\end{aligned}$$

- notes:
  - single trusted party is replaced with two parties trusted by one entity each
  - if f and g are not colluding, then they cannot compute kuv
  - both u and v trust at least one of f and g for not colluding

Pace of establishment of the security associations

- **Depends on several factors:**
  - Area size
  - Number of communication partners: *s*
  - Number of nodes: *n*
  - Number of friends
  - Mobility model and its parameters (speed, pause times, …)

Desired security associations :
p_{ij} = 1, if i and j wants to setup a shared key, and 0 otherwise

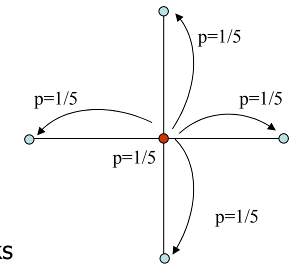
Established security associations :
$e_{ij}(t)$ = 1, if at time t nodes i and j already share a key, and 0 otherwise

Convergence : $$r(t) = \frac{\sum_{i,j} e_{ij}(t) \cdot p_{ij}}{\sum_{i,j} p_{ij}}$$

and the convergence time t_M is the earliest time at which $r(t) = 1$.

Mobility models

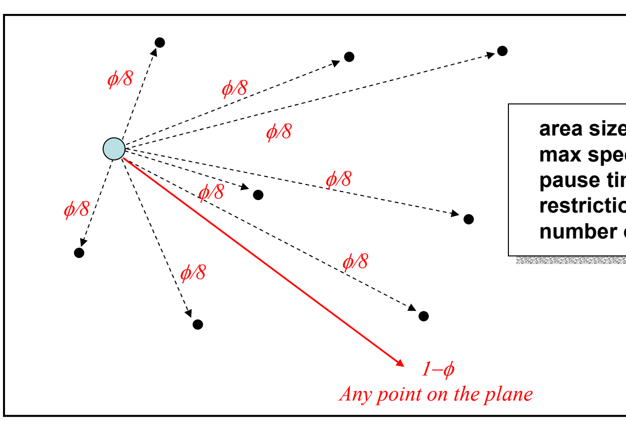
- Random walk
  - discrete time
  - simple, symmetric random walk
  - **area:** Bounded and toroid grids (33x33, 100x100, 333x333)
  - **number of nodes:** 100



- Random waypoint
  - most commonly used in mobile ad hoc networks
  - continuous time
  - **area size:** 1000m x1000m
  - **max speed:** 5m/s, 20m/s
  - **pause time:** 5s, 100s, 200s
  - **security power range**: 5m (SSC), 50m 100m (radio)
- Common simulation settings
  - simulations are run 20 times
  - confidence interval: 95%

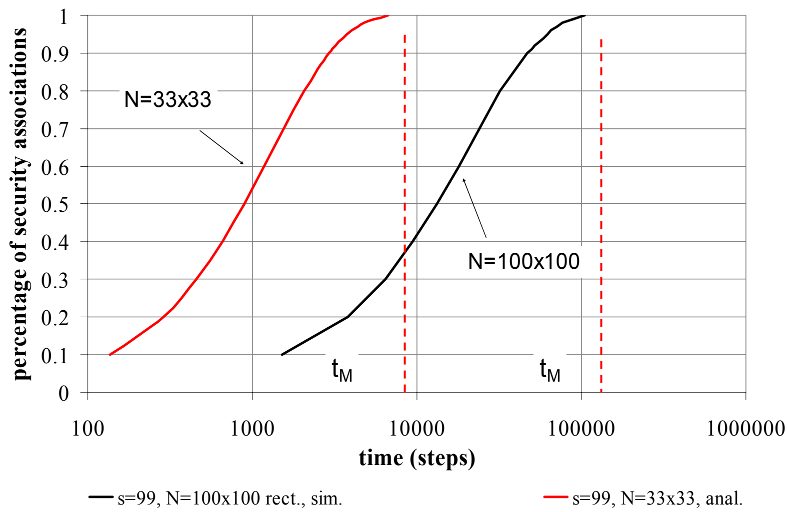
(Restricted) random waypoint

- Restricts the movement of nodes to a set of points with a predefined probability ϕ
- Regular random waypoint is a special case (ϕ = 0)



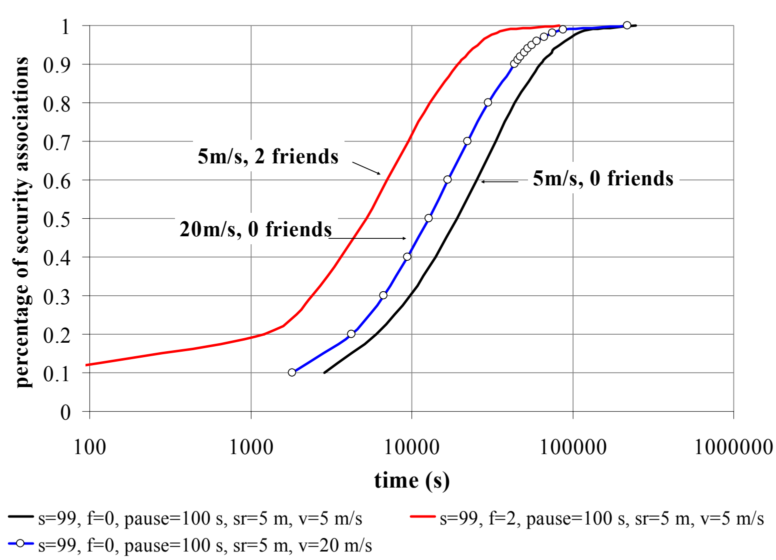
area size: 1000m x1000 m
max speed: 5m/s, 20m/s
pause time: 5s, 100s, 200s
restriction probability: 0.1, 0.5, 1
number of restriction points: 20

Size matters

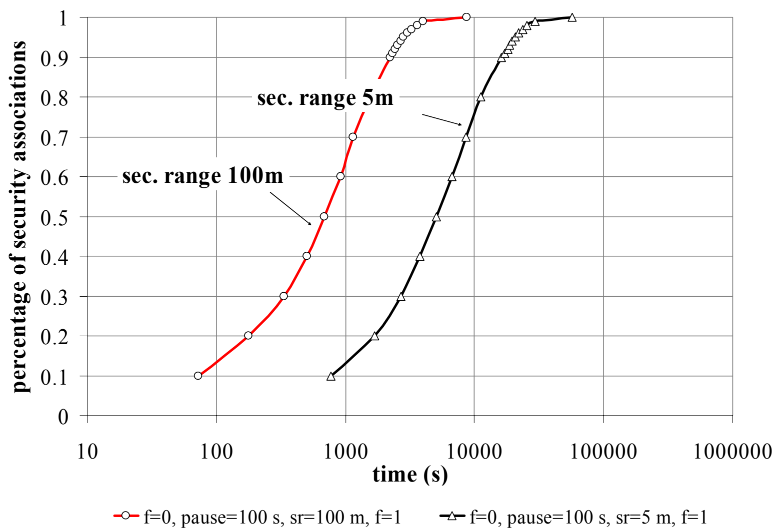


$t_M = O(N \log N)$

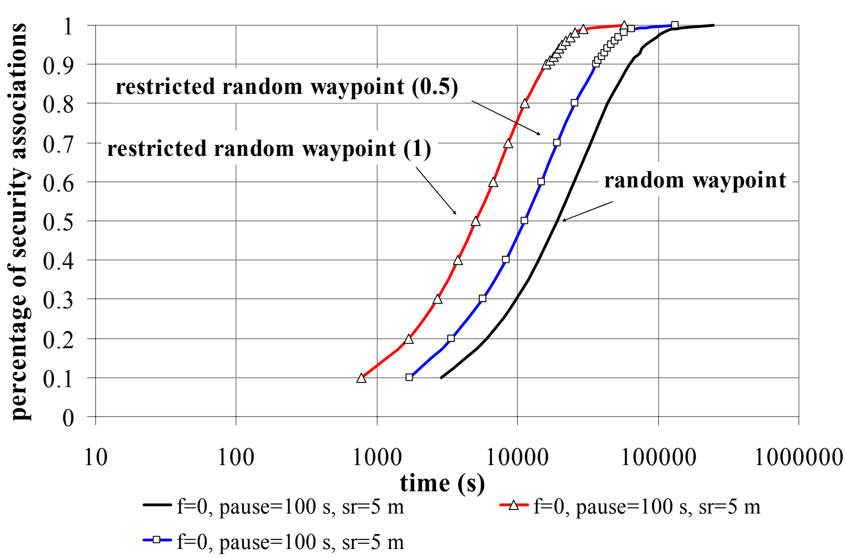
Friends help



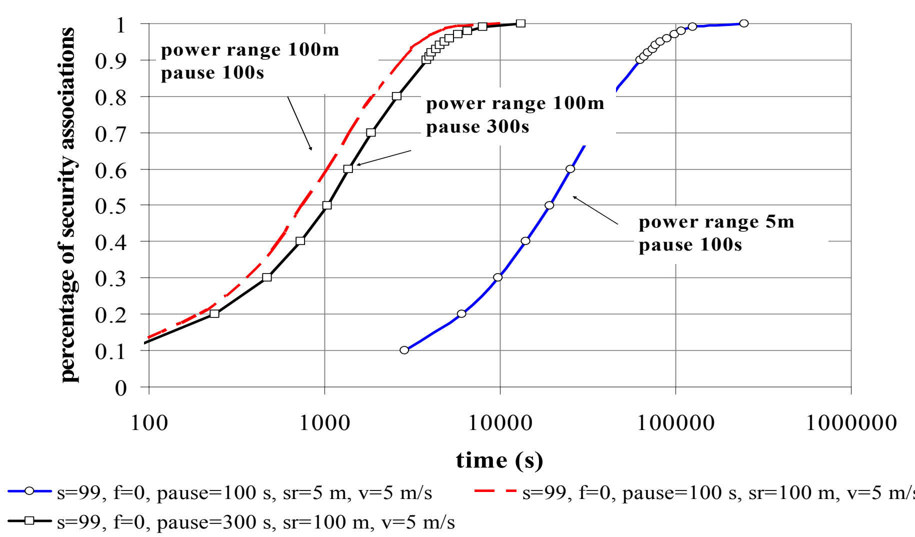
Security range matters



Meeting points help



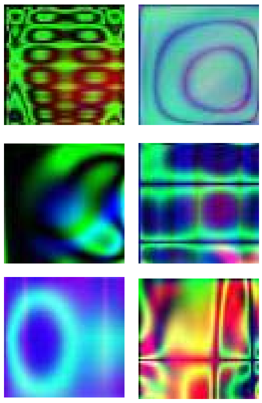
Pause time



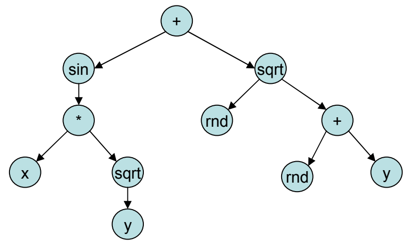
Can the secure side channel be based on radio?

→ Diffie-Hellman with authentication of messages by the human operator of the devices
  - Alice and Bob comes together
  - their devices execute the basic Diffie-Hellman protocol over the radio channel
  - Alice's device hashes the concatenation of all messages that it sent and that it received in the protocol
  - Bob's device hashes the concatenation of all messages that it received and that it sent in the protocol
  - the two hash must match
  - both devices display the hash values in some human readable format
    - select a set of worlds from a common dictionary based on the hash value
    - visualize the hash function with a Random Art image
  - the human operators compare the displayed hash value

Application of Random Art images



- the hash value is converted into three two-variable functions
  - each can be represented as a tree



- these formulae define the color value of each pixel of the resulting image

Can the secure side channel be based on radio?

→ Diffie-Hellman with I-codes

- special PHY layer that uses on-off keying (presence of signal means 1, absence of signal means 0)
- active adversary can turn a 0 into a 1, but not vice versa
- each message sent is encoded in such a way that it contains the same number of 0s and 1s (I-codes)
- messages cannot be modified in an unnoticeable way
- encoding messages with I-codes ensures the integrity of the communications → Man-in-the-Middle is excluded

Summary

- it is possible to establish pairwise shared keys in ad hoc networks without a globally trusted third party
- mobility, secure side channels, and friends are helpful
- the pace of establishment of the security associations is strongly influenced by the area size, the number of friends, and the speed of the nodes
- the proposed solution can easily be implemented with both symmetric and asymmetric crypto

Security and Privacy in Upcoming Wireless Networks

Key establishment in sensor networks

key types;

establishment of link keys using a short-term master key;

random key pre-distribution:

- the basic scheme, and
- some improvements;

Key establishment in sensor networks

- due to resource constraints, asymmetric key cryptography should be avoided in sensor networks
- we aim at setting up symmetric keys

- requirements for key establishment depend on
  - communication patterns to be supported
    - unicast
    - local broadcast
    - global broadcast
  - need for supporting in-network processing
  - need to allow passive participation

- useful key types
  - node keys – shared by a node and the base station
  - link keys – pairwise keys shared by neighbors
  - cluster keys – shared by a node and all its neighbors
  - network key – a key shared by all nodes and the base station

Setting up node, cluster, and network keys

- node key
  - can be preloaded into the node before deployment

- cluster key
  - can be generated by the node and sent to each neighbor individually protected by the link key shared with that neighbor

- network key
  - can also be preloaded in the nodes before deployment
  - needs to be refreshed from time to time (due to the possibility of node compromise)
    - neighbors of compromised nodes generate new cluster keys
    - the new cluster keys are distributed to the non-compromised neighbors
    - the base station generates a new network key
    - the new network key is distributed in a hop-by-hop manner protected with the cluster keys

Design constraints for link key establishment

- network lifetime
  - severe constraints on energy consumption
- hardware limits
  - 8-bit CPU, small memory
  - large integer arithmetics are infeasible
- no tamper resistance
  - nodes can be compromised
  - secrets can be leaked
- no a priori knowledge of post-deployment topology
  - it is not known a priori who will be neighbors
- gradual deployment
  - need to add new sensors after deployment

Traditional approaches

- use of public key crypto (e.g., Diffie-Hellman)
  - limited computational and energy resources of sensors
- use of a trusted key distribution server (Kerberos-like)
  - base station could play the role of the server
  - requires routing of key establishment messages to and from the base station
    - routing may already need link keys
    - unequal communication load on the sensors
  - base station becomes single point of failure
- pre-loaded link keys in sensors
  - post-deployment topology is unknown
  - single "mission key" approach
    - vulnerable to single node compromise
  - n-1 keys in each of the n sensors
    - excessive memory requirements
    - gradual deployment is difficult
    - doesn't scale

Link key setup using a short-term master key (LEAP)

- main assumptions:
  - any sensor node deployed will not be compromised within T_{min} time
  - any node can discover its neighbors and set up neighbor relationships within $T_{est} < T_{min}$ time
    - typically, T_{est} is a few seconds, so these assumptions make sense in practice

- protocol:
  - key pre-distribution phase
  - neighbor discovery phase
  - link key establishment phase
  - key erasure phase

Link key setup using a short-term master key (LEAP)

- key pre-distribution phase
  - before deployment, each node is loaded with K_I
  - each node u derives a node key K_u as $K_u = f(K_I, u)$, where f is a one-way function

- neighbor discovery phase
  - when a node deployed, it tries to discover its neighbors by broadcasting a HELLO message

$$u \rightarrow *: u, N_u$$

    where N_u is a random nonce
  - each neighbor v replies with

$$v \rightarrow u: v, mac(K_v, v|N_u)$$

  - u can compute $f(K_I, v) = K_v$, and verify the authenticity of the reply

Link key setup using a short-term master key (LEAP)

- link key establishment phase
  - u computes the link key K_{uv} = f(K_v, u)
  - v computes the same key
  - no messages are exchanged
  - note:

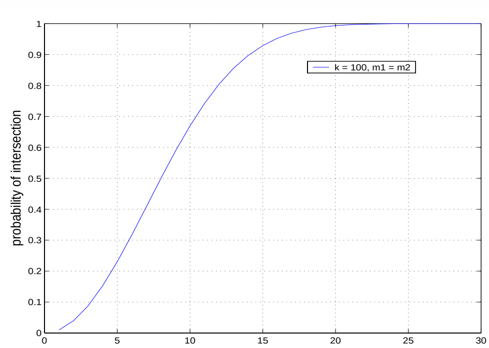
    u does not authenticate itself to v, but …
    - only a node that knows K_I can compute K_{uv}
    - a compromised node that tries to impersonate u cannot know K_I (see below)

- key erasure phase
  - T_{min} time after its deployment, each node deletes K_I and all keys it computed in the neighbor discovery phase

Random key pre-distribution – Preliminaries

Given a set *S* of *k* elements, we randomly choose two subsets S_1 and S_2 of m_1 and m_2 elements, respectively, from *S*.
The probability of $S_1 \cap S_2 \neq \emptyset$ is

$$\Pr\{S_1 \cap S_2 \neq \emptyset\} = 1 - \frac{(k-m_1)!(k-m_2)!}{k!(k-m_1-m_2)!}$$



The basic random key pre-distribution scheme

- initialization phase
  - a large pool S of unique keys are picked at random
  - for each node, m keys are selected randomly from S and pre-loaded in the node (key ring)
- direct key establishment phase
  - after deployment, each node finds out with which of its neighbors it shares a key (e.g., each node may broadcast the list of its key IDs)
  - two nodes that discover that they share a key verify that they both actually posses the key (e.g., execute a challenge-response protocol)
- path key establishment phase
  - neighboring nodes that do not have a common key in their key rings establish a shared key through a path of intermediaries
  - each link of the path is secured in the direct key establishment phase

Setting the parameters

- connectivity of the graph resulting after the direct key establishment phase is crucial
- a result from random graph theory [Erdős-Rényi]:

  in order for a random graph to be connected with probability *c* (e.g., *c* = 0.9999), the expected degree *d* of the vertices should be:

$$d = \frac{n-1}{n}(\ln(n) - \ln(-\ln(c))) \qquad (1)$$

- in our case, *d* = *pn'* (2), where *p* is the probability that two nodes have a common key in their key rings, and *n'* is the expected number of neighbors (for a given deployment density)
- *p* depends on the size *k* of the pool and the size *m* of the key ring

$$p = 1 - \frac{((k-m)!)^2}{k!(k-2m)!} \qquad (3)$$

- $c \xrightarrow{(1)} d \xrightarrow{(2)} p \xrightarrow{(3)} k, m$

Setting the parameters – an example

- number of nodes: $n = 10000$
- expected number of neighbors: $n' = 40$
- required probability of connectivity after direct key establishment: $c = 0.9999$

- using (1):
  required node degree after direct key establishment: $d = 18.42$
- using (2):
  required probability of sharing a key: $p = 0.46$
- using (3):
  appropriate key pool and key ring sizes:
  $k = 100000$, $m = 250$
  $k = 10000$, $m = 75$
  …

Qualitative analysis

- advantages:
  - parameters can be adopted to special requirements
  - no need for intensive computation
  - path key establishment have some overhead …
    - decryption and re-encryption at intermediate nodes
    - communication overhead
  - but simulation results show that paths are not very long (2-3 hops)
  - no assumption on topology
  - easy addition of new nodes

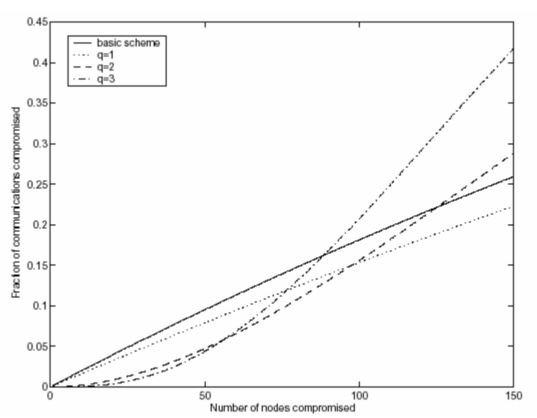
- disadvantages:
  - node capture affects the security of non-captured nodes too
    - if a node is captured, then its keys are compromised
    - these keys may be used by other nodes too
  - if a path key is established through captured nodes, then the path key is compromised
  - no authentication is provided

Improvements: q-composite rand key pre-distribution

- basic idea:
  - two nodes can set up a shared key if they have at least *q* common keys in their key rings
  - the pairwise key is computed as the hash of all common keys
- advantage:
  - in order to compromise a link key, all keys that have been hashed together must be compromised
- disadvantage:
  - probability of being able to establish a shared key directly is smaller (it is less likely to have *q* common keys, than to have one)
  - key ring size should be increased (but: memory constraints) or key pool size should be decreased (but: effect of captured nodes)

q-composite scheme: Simulation results

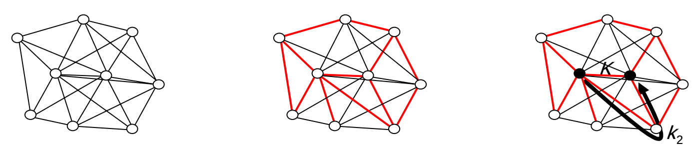
$m = 200$, $p = 0.33$



taken from: H. Chan and A. Perrig and D. Song, "Random key predistribution schemes for sensor networks", IEEE Security and Privacy Symp. (Oakland), 2003

Improvements: Multipath key reinforcement

- basic idea:
  - establish link keys through multiple disjoint paths
  - assume two nodes have a common key K in their key rings
  - one of the nodes sends key shares $k_1, \ldots, k_j$ to the other through j disjoint paths
  - the key shares are protected during transit by keys that have been discovered in the direct key establishment phase
  - the link key is computed as $K + k_1 + \ldots + k_j$



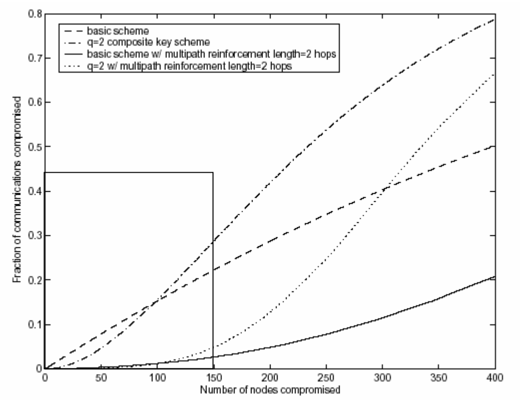
radio connectivity   shared key connectivity   multipath key reinforcement

Improvements: Multipath key reinforcement

- advantages:
  - in order to compromise a link key, at least one link on each path must be compromised → increased resilience to node capture
- disadvantages:
  - increased overhead
- note:
  - multipath key reinforcement can be used for path key setup too

Multipath scheme: Simulation results

$m = 200$, $p = 0.33$



taken from: H. Chan and A. Perrig and D. Song, "Random key predistribution schemes for sensor networks", IEEE Security and Privacy Symp. (Oakland), 2003

Polynomial based key pre-distribution

- let f be a bivariate t-degree polynomial over a finite field GF(q), where q is a large prime number, such that f(x, y) = f(y, x)

$$f(x, y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j$$

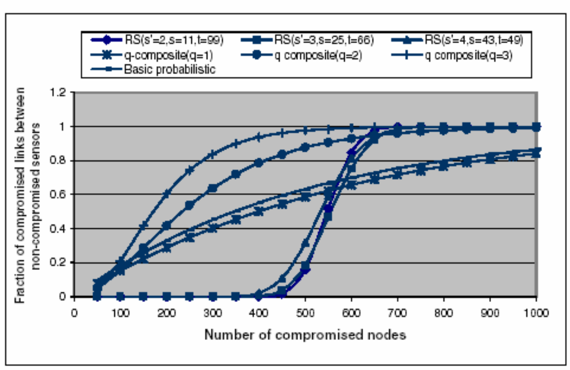
- each node is pre-loaded with a polynomial share f(i, y), where i is the ID of the node
- any two nodes i and j can compute a shared key by
  - i evaluating f(i, y) at point j and obtaining f(i, j), and
  - j evaluating f(j, y) at point i and obtaining f(j, i) = f(i, j)
- this scheme is unconditionally secure and t-collusion resistant
  - any coalition of at most t compromised nodes knows nothing about the shared keys computed by any pair of non-compromised nodes
- any pair of nodes can establish a shared key without communication overhead (if they know each other's ID)
- memory requirement of the nodes is (t +1) log(q)
- problem: t is limited by the memory constraints of the sensors

Polynomial based random key pre-distribution

- operation:
  - let S be a pool of bivariate t-degree polynomials
  - for each node i, we pick a subset of m polynomials from the pool
  - we pre-load into node i the polynomial shares of these m polynomials computed at point i
  - two nodes that have polynomial shares of the same polynomial f can establish a shared key f(i, j)
  - if two nodes have no common polynomials, they can establish a shared key through a path of intermediaries
- advantage:
  - can tolerate the capture of much more than t nodes (t can be smaller, but each node needs to store m polynomials)
    - in order to compromise a polynomial, the adversary needs to obtain t + 1 shares of that polynomial
    - it is very unlikely that t + 1 randomly captured nodes have all selected the same polynomial from the pool

Simulation results

- $m = 200$, $p = 0.33$



taken from D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks", ACM CCS, 2003.

Matrix based key pre-distribution (Blom's scheme)

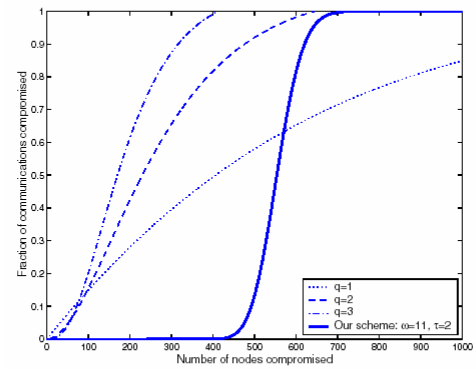
- let G be a $(t + 1)\times n$ matrix over a finite field GF(q) (where n is the size of the network)
- let D be a random $(t +1)\times(t +1)$ symmetric matrix over GF(q)
- G is public, D is secret
- let $A = (DG)^T$ and $K = AG$
  - K is a symmetric matrix, because $K = AG = (DG)^T G = G^T D^T G = G^T DG = G^T A^T = (AG)^T = K^T$
- each node i stores the i-th row of A
- any two nodes i and j can compute a shared key K_{ij}
  - i computes $A(i,.)G(.,j) = K_{ij}$
  - j computes $A(j,.)G(.,i) = K_{ji} = K_{ij}$

Matrix based random key pre-distribution

- G is as before
- $D_1, \ldots, D_k$ are random $(t +1)\times(t +1)$ symmetric matrices
- $A_v = (D_v G)^T$ and $\{A_v\}$ is the pool (of spaces)
- for each node i, we pick a random subset of the pool and pre-load in the node the i-th row of the selected matrices (i.e., $A_v(i,.)$ for each selected v)
- if two nodes i and j both selected a common matrix A_v, then they can compute a shared key using Blom's scheme
- if two nodes don't have a common space, they can setup a key through intermediaries

Simulation results

m = 200, *p* = 0.33



taken from W. Du and J. Deng and Y. S. Han and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks", ACM CCS, 2003

Assumptions revisited (and more) [DiPietro et al.]

- the connectivity graph resulting after the first phase of the basic protocol is not a random graph (in the sense of Erdős-Rényi)
  - an edge between two nodes exists if they have at least one common key AND they are in power range of each other
  - the Erdős-Rényi theorem may not hold for these types of graphs
- almost sure connectivity after the first phase of the basic protocol is ensured if
  - $m^2 / k = c \log(n) / n$
  - $k > n-1$
  - $m > 4$

Assumptions revisited (and more) [DiPietro et al.]

- a definition of security:

  a network is *redoubtable*, if the probability that compromising a sub-linear fraction of nodes (uniformly chosen at random) can compromise a constant fraction of links in the network is 0

- a network is redoubtable if $m/k \sim 1/n$

- a practical choice of parameters:
  - $m = \log(n)$
  - $k = n \log(n)$

Summary

- in sensor networks, we need different types of keys

- node keys, cluster keys, and network keys can be established relatively easily using the technique of key pre-loading and using already established link keys

- link keys can be established using a short-term master key or with the technique of random key pre-distribution