Protection of Communication Infrastructures

Chapter 8
Security in Wireless Sensor Networks

- Introduction
- Denial of Service & Routing Security
- Energy Efficient Confidentiality and Integrity
- Authenticated Broadcast
- Alternative Approaches to Key Management
- Secure Data Aggregation

Wireless Sensor Network Characteristics (1)

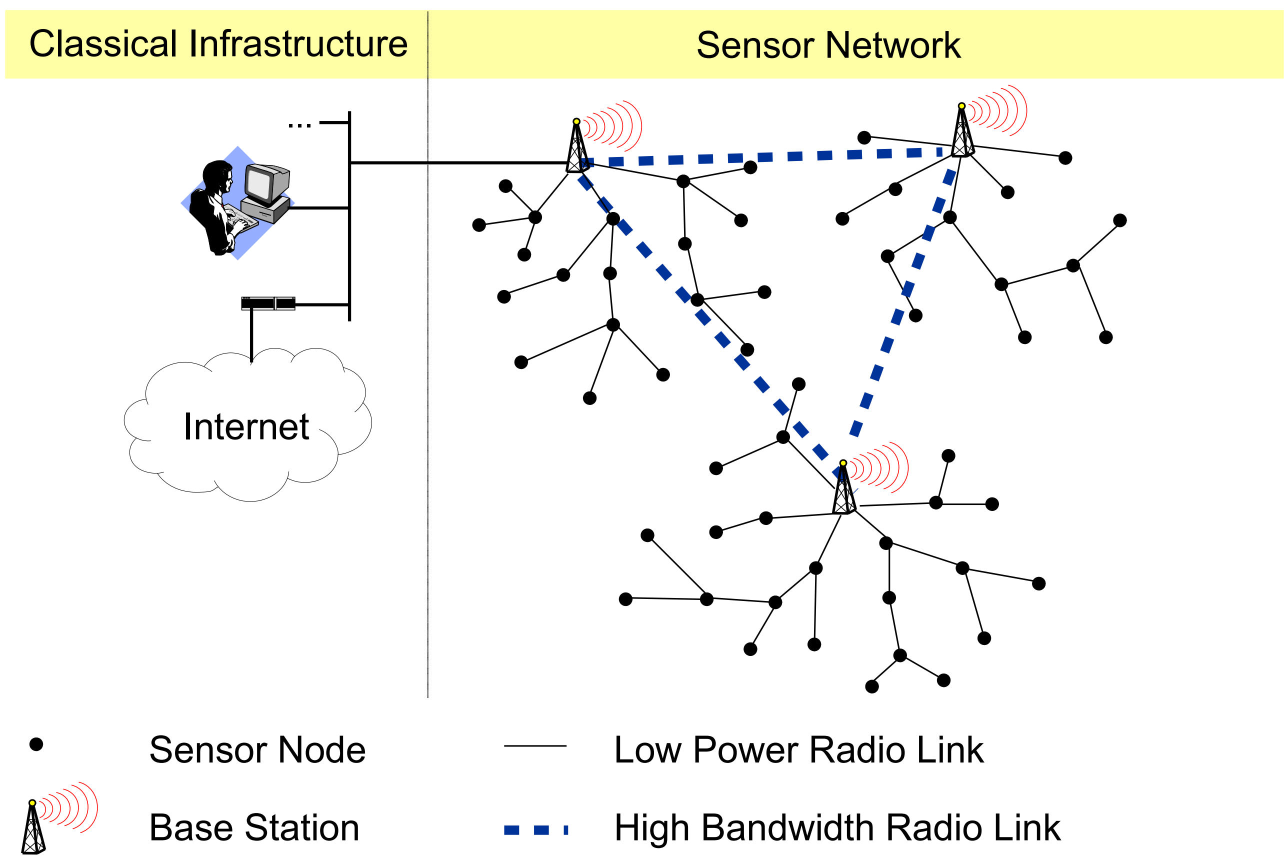
- Wireless sensor networks are envisaged to be:
  - formed by tens to thousands of small, inexpensive sensors that communicate over a wireless interface;
  - connected via base stations to traditional networks / hosts running applications interested in the sensor data;
  - using multi-hop communications among sensors in order to bridge the distance between sensors and base stations;
  - considerably resource constrained due to limited energy availability.
- Example Sensor Node:
  - 4 MHz clock
  - 8-bit processor
  - 4 KB free of 8 KB flash
  - 512 bytes SRAM
  - 19.2 Kbps radio
  - Battery-powered

[W01a]

Wireless Sensor Network Characteristics (2)

- Typical applications:
  - Environment monitoring: earthquake or fire detection, etc.
  - Home monitoring and convenience applications
  - Site surveillance: intruder detection
  - Logistics and inventory applications: tagging & locating goods, containers, ...
  - Military applications: battleground reconnaissance, troop coordination, ...
- Typical communication pattern:
  - an application demands some named information in a specific geographical area;
  - one or more base stations broadcast the request;
  - wireless sensors relay the request and generate answers to it if they contribute to the requested information;
  - answers are processed and aggregated as they flow through the network towards the base station(s).

Example Sensor Network Topology



Sensor Networks vs. Ad hoc Networks

- *Application specific characteristics*, e.g. depending on application networks might be very sparse or dense
- *Environment interaction* may cause rather bursty traffic patterns, e.g. due to incident detection
- *Scale* is expected to vary between tens to thousands of sensors
- *Energy* is even more scarce as sensors will be either battery-powered or powered by environmental phenomena (e.g. vibration)
- *Self-configurability*, as in ad hoc networks but likely to be different, e.g. human interaction prohibitive, geographic position has to be learned, ...
- *Dependability and QoS*, classical QoS notion like throughput, jitter, etc. are of little use here, what counts is delivery of requested information
- *Data centric model*, sensor identities are of little interest; new addressing schemes (semantic, geographic, ...) are more interesting
- *Simplicity* in terms of OS, networking SW, memory footprint, ...

(according to [KW03a])

Challenging Security Objectives in Wireless Sensor Networks

- Avoiding and coping with sensor node compromise:
  - Protecting sensor nodes from compromise (tamper proofing)
  - Graceful degradation in case of single node compromise
- Availability of sensor network services:
  - Robustness against Denial of Service (DoS) attacks
  - Protection of sensor nodes from malicious energy draining
  - Correct functioning of message routing
- Confidentiality and integrity of data:
  - Data retrieved from sensor networks should be protected from eavesdropping and malicious manipulation
  - This also requires an appropriate key management
- What makes these objectives particularly challenging?
  - Severe resource constraints (memory, time, energy)
  - "Unfair" power balance: powerful attackers against weak sensors
  - Different communication pattern (incl. aggregation) opts against pure end-to-end security approaches

DoS Threats & Countermeasures in Sensor Networks

Network Layer	Attacks	Countermeasures
Physical	Tampering	Tamper-proofing, hiding
	Jamming	Spread-spectrum, priority messages, lower duty cycle, region mapping, mode change
Link	Collision	Error-correcting code
	Exhaustion	Rate limitation
	Unfairness	Small frames
Network	Neglect and greed	Redundancy, Probing
	Homing	Encryption (only partial protection)
	Misdirection	Egress filtering, authorization, monitoring
	Black holes	Authorization, monitoring, redundancy
Transport	Flooding	Client puzzles
	De-synchronization	Data Origin Authentication

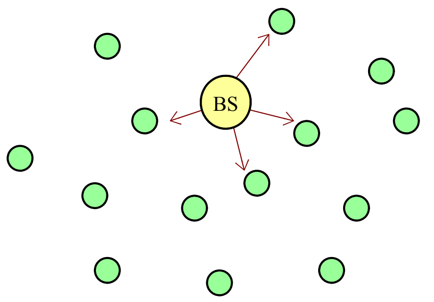
(according to [WS02a])

Sensor Network Routing Threats

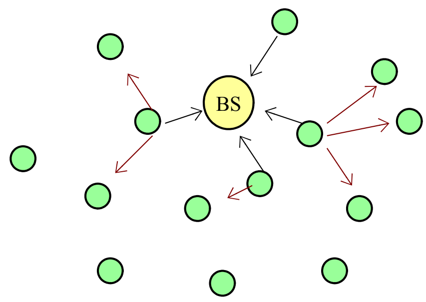
- *Spoofed, altered or replayed routing information:* may be used for loop construction, attracting or repelling traffic
- *Acknowledgement forging:* may trick other nodes to believe that a link or node is either dead or alive
- *Selective forwarding:* either "in-path" or "beneath path" by deliberate jamming, allows to control which information is forwarded
- *Sinkhole attacks:* attracting traffic to a specific node, e.g. to prepare selective forwarding
- *Simulating multiple identities ("Sybil attacks"):* allows to reduce effectiveness of fault-tolerant schemes like multi-path routing
- *Wormhole attacks:* tunneling of messages over alternative low-latency links, e.g. to confuse the routing protocol, create sinkholes. etc.
- *Hello floods (more precise: "Hello shouting"):* an attacker sends or replays a routing protocol's hello packets with more energy

[KW03a]

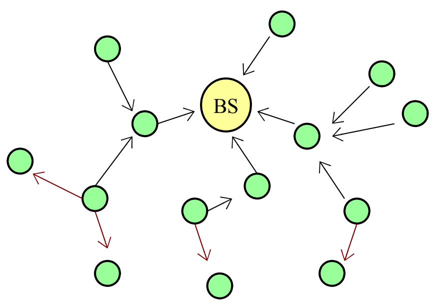
Example: Breadth First Spanning Tree



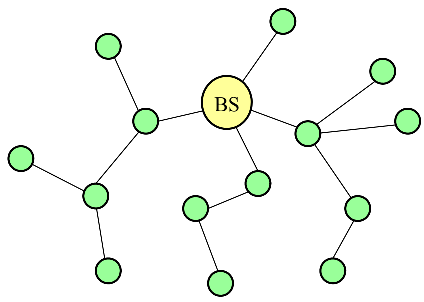
1. BS sends beacon



2. First answers to beacon



3. Answers to first answers



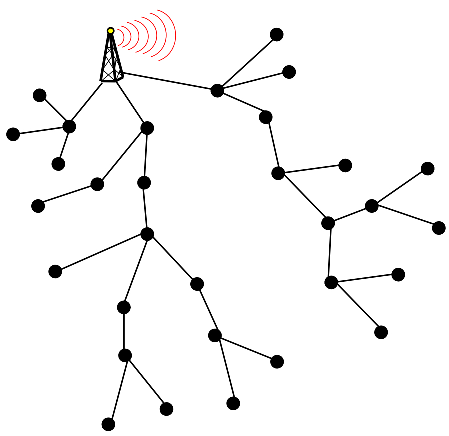
4. Resulting Routing Tree

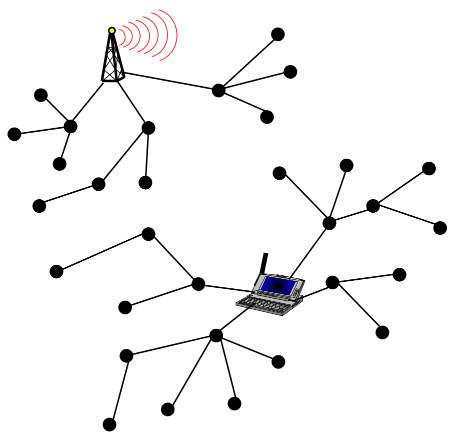
[W01a]

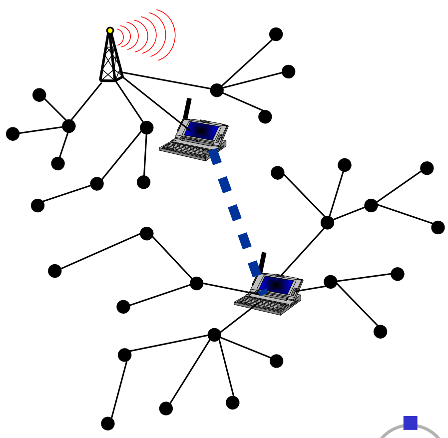
Example: Attacks on Breadth-First Spanning Tree

- TinyOS builds a breadth-first spanning tree rooted at the base station
- An attacker disposing of one or two laptops can either send out forged routing information or launch a wormhole attack
- Both attacks lead to entirely different routing trees and can be used to prepare further attacks like selective forwarding, etc.

Example Routing Tree	Forging Routing Updates	Wormhole Attack







Potential Countermeasures Against Attacks On Routing

- *Forging of routing information or acknowledgements* can be countered by data origin authentication and confidentiality of link layer PDUs:
  - First idea: use of a single group key (considered vulnerable, e.g. a single node compromise results in complete failure)
  - Second approach: Each node shares a secret key with a base station, base station acts as trusted third party in key negotiation (e.g. Otway-Rees)
- *Simulating multiple identities*: by reducing the number of neighbors a node is allowed to have – e.g. through enforcement during key distribution – the threat potential can be limited
- *Hello shouting* and *wormhole/sinkhole attacks* can not be completely countered with link layer security services:
  - Links should be checked in both directions before making routing decisions
  - Detection of wormholes requires tight clock synchronization [HPJ02a]
  - Sinkholes might be avoided with geographical routing
- *Selective forwarding* might be countered with multi-path routing

Ensuring Data Confidentiality and Integrity

- Main challenges:
  - Tight implementation constraints (instruction set, memory, speed)
  - Very small energy budget in low-powered devices (e.g. by battery)
  - Some nodes might get compromised
- The mentioned constraints opt out some well established alternatives:
  - Asymmetric cryptography is generally considered to be too expensive:
    - High computational cost + long ciphertexts/signatures (sending and receiving is very expensive!)
    - Especially, public key management based on certificates exceeds node's energy budget, key revocation almost impossible to realize
  - Even symmetric cryptography implementation might be difficult due to architectural limitations and energy constraints
  - Key management for authenticating broadcast-like communications calls for new approaches
- Exemplary approach SPINS [PS+02a]:
  - *SNEP:* for realizing end-to-end security between nodes and base stations
  - *μTESLA:* for authenticating broadcast communications

Sensor Network Encryption Protocol (1)

- Main Goal:
  - Efficient end-to-end security services for two party communication
- Security services provided:
  - Data confidentiality
  - Data origin authentication
  - Replay protection
- Considered communication patterns:
  - Node to base station, e.g. sensor readings
  - Base station to individual nodes, e.g. specific requests
  - Base station to all nodes, e.g. routing beacons, queries, re-programming of the entire network (secured with μ*TESLA)*
- Design decisions:
  - No use of asymmetric cryptography
  - Construct all cryptographic primitives out of a single block cipher
  - Exploit common state to reduce communication overhead

Sensor Network Encryption Protocol (2)

- Basic trust model and key derivation:
  - Two communicating entities A and B share a common master key $X_{A,B}$:
    - Initially, the base station shares a master key with all nodes
    - Node-to-node keys can be negotiated with help of the base station
  - Four session keys and a random seed are derived from this master key:
    - Confidentiality keys: $CK_{A,B} := F_{X_{A,B}}(1)$
      $CK_{B,A} := F_{X_{A,B}}(3)$
    - Integrity keys: $IK_{A,B} := F_{X_{A,B}}(2)$
      $IK_{B,A} := F_{X_{A,B}}(4)$
    - Random generator seed: $RK_{A,B} := F_{X_{A,B}}(5)$
- Principal cryptographic primitive is the RC5 algorithm:
  - Configurable parameters: word length w [bit], number of rounds r, key size b [byte], denoted as RC5-w/r/b
  - Operations:
    - Two's complement addition of words (mod 2w) $+$
    - Bit-wise XOR of words $\oplus$
    - Cyclic rotation <<<
  - Key Setup: an array S[0, 2r + 1] of words is filled by a setup procedure

RC5 Encryption and Security

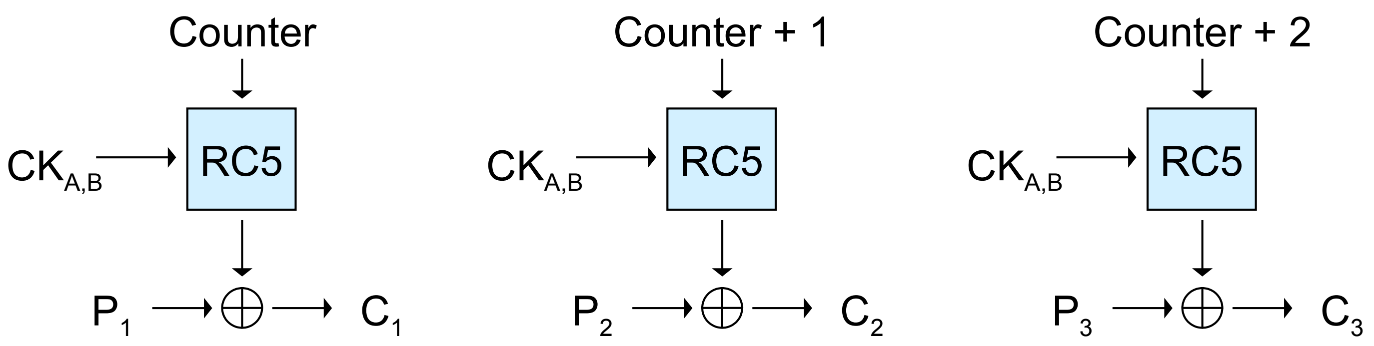
- Encryption function with plaintext / ciphertext in two words A, B:
  - A := A + S[0];
    B := B + S[1];
    for i := 1 to r
        A := ((A ⊕ B) <<< B) + S[2i];
        B := ((B ⊕ A) <<< A) + S[2i + 1];

Plaintext Requirements for Differential Attacks (Block Size 64)

Number of Rounds	4	6	8	10	12	14	16
Differential Attack (Chosen Plaintext)	2^{7}	2^{16}	2^{28}	2^{36}	2^{44}	2^{52}	2^{61}
Differential Attack (Known Plaintext)	2^{36}	2^{41}	2^{47}	2^{51}	2^{55}	2^{59}	2^{63}

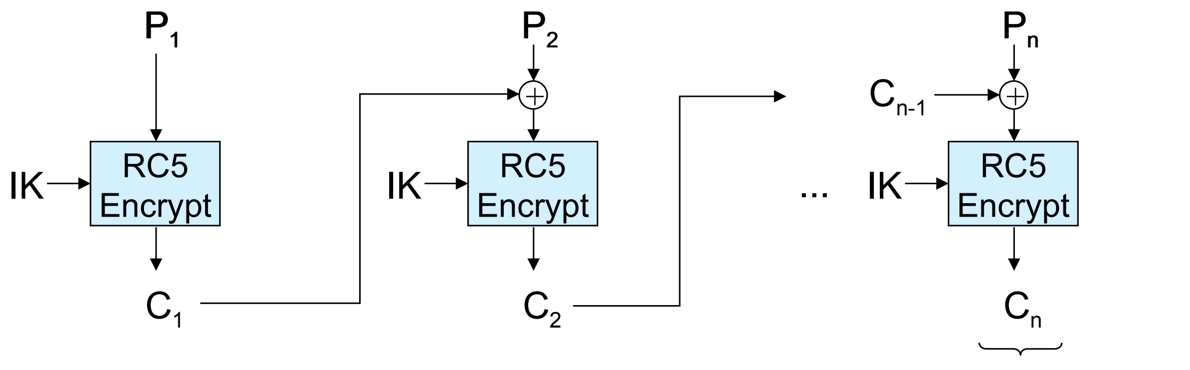
[KY98a]

SNEP Confidentiality: RC5 Encryption in Counter Mode



- RC5 generates pseudo-random bit stream to XOR with plaintext:
  - Ciphertext is denoted as $\{P\}_{<K_{A,B},\ Counter>}$
  - In order to decrypt a ciphertext, the same pseudo-random stream is generated and XORed with the ciphertext
- Counter is shared state and may never be reused with same key to encrypt two (or more) different plaintexts:
  - Otherwise, an attacker can obtain the XOR of the two plaintexts by XORing the respective ciphertexts

SNEP Integrity: RC5 Cipher Block Chaining MAC



- SNEP uses RC5-CBC MAC (message authentication code)
- Two message formats (without or with confidentiality):
  - $A \rightarrow B$: Msg | RC5-CBC($IK_{A,B}$, Msg)
  - $A \rightarrow B$: $\{Msg\}_{<CK_{A,B}, Counter>}$ | RC5-CBC ($IK_{A,B}$, Counter | $\{Msg\}_{<CK_{A,B}, Counter>}$)
- Further cryptographic issues:
  - RC5-CBC is also used for key derivation: $F_{\chi_{A,B}}(n)$:= RC5-CBC($\chi_{A,B}$, n)
  - Random numbers are generated by encrypting a counter

SNEP Counter Synchronization and Replay Protection

- Counter synchronization:
  - Initial counter negotiation:
    - $A \rightarrow B$: C_A
    - $B \rightarrow A$: C_B | RC5-CBC($IK_{B,A}$, C_A | C_B)
    - $A \rightarrow B$: RC5-CBC($IK_{A,B}$, C_A | C_B)
  - Message losses might be handled by trying different counters:
    - consumes energy $\Rightarrow$ only a few counter values can be tried
  - If counters get out of synch, explicit re-synchronization is carried out:
    - $A \rightarrow B$: N_A // N_A denoting a fresh random number generated by A
    - $B \rightarrow A$: C_B | RC5-CBC($IK_{B,A}$, N_A | C_B)
- Replay Protection:
  - Encrypted messages have implicit replay protection provided by the counter used in RC5 encryption
  - For tighter time synchronization, a nonce based dialog can be used:
    - $A \rightarrow B$: N_A | Req
    - $B \rightarrow A$: $\{Rsp\}_{<CK_{B,A}, CB>}$ | RC5-CBC($IK_{B,A}$, N_A | C_B | $\{Rsp\}_{<CK_{B,A}, CB>}$)

SNEP Node-to-Node Key Agreement

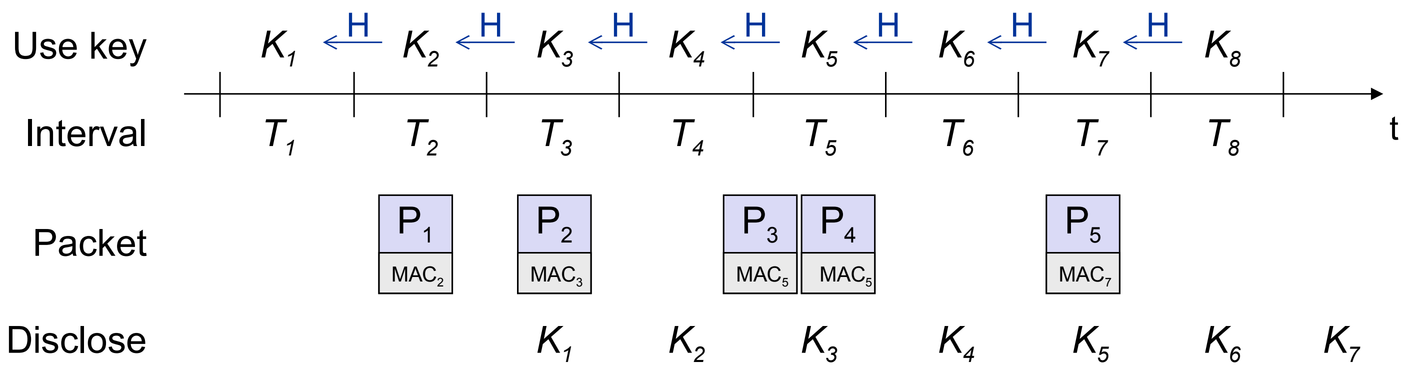
- In order to establish a shared secret $SK_{A,B}$ between A and B with the help of base station BS, the following protocol is proposed:
  - $A \rightarrow B$: N_A | A
  - $B \rightarrow BS$: N_A | N_B | A | B | RC5-CBC($IK_{B,BS}$, N_A | N_B | A | B)
  - $BS \rightarrow A$: $\{SK_{A,B}\}_{K_{BS,A}}$ | RC5-CBC($IK_{BS,A}$, N_A | B | $\{SK_{A,B}\}_{K_{BS,A}}$)
  - $BS \rightarrow B$: $\{SK_{A,B}\}_{K_{BS,B}}$ | RC5-CBC($IK_{BS,B}$, N_B | A | $\{SK_{A,B}\}_{K_{BS,B}}$)
- Discussion:
  - The session key $SK_{A,B}$ is generated by the base station
  - The random numbers N_A and N_B shall provide assurance of freshness
  - However, the protocol does neither allow A nor B to perform concurrent key negotiations with multiple entities
  - Neither A nor B knows, if the other party received the key and trusts in its suitability
  - The base station does not know about the freshness of messages

Authenticated Broadcast with μTESLA

- Requirements:
  - Must have asymmetric mechanism to prevent forgery from recipients
  - Classical asymmetric digital signatures are too expensive in terms of computation, storage, and communication
- Basic idea for obtaining asymmetry:
  - Delayed key disclosure
  - Requires loosely synchronized clocks
- Original TESLA and μTESLA:
  - TESLA stands for *Timed Efficient Stream Loss-tolerant Authentication*
  - Principal idea is inverse use of hash-chains for obtaining integrity keys (basically, a variation of the one-time password idea)
  - μTESLA is a minor variant of TESLA:
    - TESLA uses asymmetric digital signatures to authenticate initial keys, μTESLA uses a protocol based on symmetric cryptography (SNEP)
    - μTESLA discloses the key only once per time interval, and only base stations authenticate broadcast packets (storage of key chains)

µTESLA Functions (1)

- Sender setup:
  - Choose length *n* of key chain and generate a random K_n
  - Compute and store hash key chain according to $K_{n-1} := H(K_n)$
- Broadcasting authenticated packets:
  - Time is divided in uniform length intervals T_i
  - In time interval T_i the sender authenticates packets with key K_i
  - The key K_i is disclosed in time interval $i + \delta$ (e.g. $\delta = 2$)



µTESLA Functions (2)

- Provision of a new receiver M with an authenticated initial key:
  - $M \rightarrow BS: N_M$
  - $BS \rightarrow M: T_{BS} \mid K_i \mid T_i \mid T_{Int} \mid \delta \mid RC5\text{-}CBC(IK_{BS,M}, N_M \mid T_{BS} \mid K_i \mid T_i \mid T_{Int} \mid \delta)$ with T_{Int} denoting the interval length
- Verification of authenticated broadcast packets:
  - Receiver must know current time, maximum clock drift and interval length
  - Packets must be stored with T_i until appropriate key is disclosed
  - Upon disclosure of the appropriate key K_i the authenticity of the packet can be checked
  - It is crucial to discard all packets that have been authenticated with an already disclosed key (requires loose time synchronization with appropriate value for δ)
- Authenticated broadcast by sensor nodes:
  - Sensor node sends a SNEP protected packet to base station
  - Base station sends an authenticated broadcast
  - Main reason: sensor nodes do not have enough memory for key chains

Some Remarks Concerning Security of SPINS

- According to the information given in [PS+02a] (RAM requirements, etc.), SNEP seems to use RC5 with 8 rounds and 32 bit words
  - This is on the edge of being secure against differential cryptanalysis [KY98a]
- SNEP's node-to-node key establishment procedure does not attain all customary security goals (e.g. mutual knowledge who holds a session key and has trust in it)
- Time synchronization is critical for µTESLA
  - Un-synchronized clocks might be exploited for forging MACs
  - Keys have to be disclosed soon after their usage, as nodes can not store many packets (⇒ requires tight synchronization)

Alternative Approaches to Key Management (1)

- Starting point – some common approaches to distributing keys do not work well in wireless sensor networks:
  - Asymmetric cryptography:
    - Requires very resource intensive computations and is, therefore, often judged as being not appropriate for sensor networks
  - Arbitrated key management based on pre-determined keys:
    - Some approaches like SPINS assume pre-determined keys at least between the base station and sensor nodes
    - This requires pre-distribution of these keys before deployment of the sensor network and also has some security implications in case of node compromise
  - What are specific requirements to sensor network key management?
- Some new alternatives to the traditional approaches listed above:
  - Neighborhood-based initial key exchange, e.g. LEAP
  - Probabilistic key distribution schemes

Alternative Approaches to Key Management (2)

- Requirements to key management schemes for sensor networks resulting from specific characteristics [CPS03]:
  - *Vulnerability of nodes to physical capture and node compromise:*
    - Nodes may be deployed in difficult to protect / hostile environments
    - Because of cost constraints, nodes will not be tamper-proof, so that cryptographic keys might be captured by an attacker
    - Therefore, compromise of some nodes should not compromise the overall network's security
  - *Lack of a-priori knowledge of deployment configuration:*
    - Some sensor networks are installed via random scattering (e.g. from an airplane), thus neighborhood relations are not known a-priori
    - Even with manual installation, pre-configuration of sensors would be expensive in large networks
    - Thus, sensor networks key management should support for "automatic" configuration after installation

Alternative Approaches to Key Management (3)

- More requirements from specific characteristics:
  - *Resource restrictions:*
    - Limited memory resources
    - Limited bandwidth and transmission power
  - *In-network processing:*
    - Over-reliance on base station as source of trust may result in inefficient communication patterns (→ aggregation)
    - Also, it turns base stations into attractive targets (which they are in any case!)
  - *Need for later addition of sensor nodes:*
    - Compromise, energy exhaustion or limited material / calibration lifetime may make it necessary to add new sensors to an existing network
    - Legitimate nodes that have been added to sensor network should be able to establish secure relationships with existing nodes
    - Erasure of master keys after initial installation (→ LEAP) does not allow this

Alternative Approaches to Key Management (4)

- Criteria for evaluating sensor network key management schemes:
  - *Resilience against node compromise:*
    - How many communication relationships are affected from the compromise of a node and the cryptographic secrets stored in it?
    - Of course, communication relationships with the compromised node itself are always affected and do not count here
  - *Resistance against node insertion / replication:*
    - Is an attacker able to insert malicious nodes in the network with legitimate looking identities?
    - Can compromised nodes be replicated (e.g. to affect voting schemes)?
  - *Revocation:*
    - Can nodes that have been detected to be compromised be revoked in the network?
  - *Scale:*
    - Does key management place restrictions on the maximum size of a sensor network?

Alternative Approaches to Key Management (5)

- The *Localized Encryption and Authentication Protocol (LEAP)* enables "automatic" and efficient establishment of security relationships in an initialization phase after installation of the nodes
- LEAP supports key establishment for various trust relationships between:
  - Base station and sensor with "individual keys"
  - Sensors that are direct neighbors with "pairwise shared keys"
  - Sensors that form a cluster with "cluster keys"
  - All sensors of a network with a "group key"
- Establishing individual keys:
  - Prior to deployment, every sensor node *u* is pre-loaded with an individual key K_u^m known only to the node and the base station
  - The base station *s* generates these keys from a master key K_s^m and the node identity u: $K_u^m := f(K_s^m, u)$
  - Generating all node keys from one master key is supposed to save memory at the base station

Alternative Approaches to Key Management (6)

- Establishing pairwise shared keys:
  - In scenarios in which pairwise shared keys cannot be pre-loaded into sensor nodes because of installation by random scattering but neighboring relationships remain static after installation, the following scheme is proposed
  - It is assumed that there is a minimum time interval T_{min} during which a node can resist against attacks
  - After being scattered, sensor nodes establish neighboring relations during this time interval based on an initial group key K_I that has been pre-configured into all sensor nodes before deployment:
    - Every node u computes its master key: $K_u = f(K_I, u)$
    - Every node discovers its neighbors by sending a message with his identity u and a nonce r_u, and collecting the answers:
      - $u \rightarrow *: u, r_u$
      - $v \rightarrow u: v, MAC(K_v, r_u \mid v)$
    - As u can also compute K_v, it can directly check this MAC
    - Both nodes compute $K_{u,v} = f(K_v, u)$

Alternative Approaches to Key Management (7)

- Establishing pairwise shared keys (cont.):
  - After expiration of timer T_{min}, all nodes erase the initial group key K_I and all computed master keys (only pairwise shared keys are kept)
  - This scheme can be augmented with all nodes forwarding also the identities of their neighbors, enabling a node also to compute pairwise shared keys with nodes that are one hop away
- Establishing cluster keys:
  - In order to establish a cluster key with all its immediate neighbors, a node randomly generates a cluster key K_u^c and sends it individually to all neighbors $v_1, v_2, \ldots$:
    - $u \rightarrow v_i: E(K_{u,vi}, K_u^c)$
    - All nodes v_i decrypt this message with their pairwise shared key $K_{u,vi}$
  - When a node is revoked, a new cluster key is distributed to all remaining nodes

Alternative Approaches to Key Management (8)

- Establishing multi-hop pairwise shared keys:
  - If a node u wants to establish a pairwise shared key with a node c that is multiple hops away, it can do so by using other nodes it knows as proxies
  - In order to detect suitable proxy nodes v_i, u broadcasts a query message with its own node id and the node id of c; nodes v_i knowing both nodes u and c will answer to this:
    - $u \rightarrow *: u, c$
    - $v_i \rightarrow u: v_i$
  - Assuming that node u has received m answers, it then generates m shares $sk_1, ..., sk_m$ of the secret key $K_{u,c}$ to be established with c and sends them individually over the appropriate nodes v_i:
    - $u \rightarrow v_i: E(K_{u,vi}, sk_i), f(sk_i, 0)$
    - $v_i \rightarrow c: E(K_{vi,c}, sk_i), f(sk_i, 0)$
  - The value $f(sk_i, 0)$ allows the nodes v_i and c to verify if the creator of such a message actually knew the key share sk_i
  - After receiving all values sk_i node c computes $K_{u,c} = sk_1 \oplus ... \oplus sk_m$

Alternative Approaches to Key Management (9)

- Establishing group keys:
  - In order to establish a new group key K_g , the base station randomly generates a new key and sends it encrypted with its own cluster key to its neighbors:
    - $s \rightarrow v_i: E(K_u^c, K_g)$
  - All nodes receiving such a message forward the new group key encrypted with their own cluster key to their neighbors
- Revoking a node:
  - Node revocation is performed by the base station and uses μTESLA
  - All nodes, therefore, have to be pre-loaded with the authentic initial key, and loose time synchronization is needed in the sensor network
  - In order to revoke a node u, the base station s broadcasts the following message in time interval T_i using the μTESLA key K_i valid for that interval:
    - $s \rightarrow *: u, f(K'_g, 0), MAC(K_i, u \mid f(K'_g, 0)),$
    - The value $f(K'_g, 0)$ later on allows all nodes to verify the authenticity of a newly distributed group key K'_g
    - This revocation becomes valid after disclosure of K_i

Alternative Approaches to Key Management (10)

- Remarks to some security aspects of LEAP:
  - As every node *u* knowing K_I may compute the master key K_v of every other node *v*, there is little additional security to be expected from distinguishing between these different "master keys":
    - Especially, all nodes need to hold K_I during the discovery phase in order to be able to compute the master keys of answering nodes
    - The authors of [ZSJ03] give no reasoning for why they think that this differentiation of master keys should attain any additional security
    - As any MAC construction that deserves its name should not leak information about K_I in a message authentication code $MAC(K_I, r_u \mid v)$, it is hard to see any benefit in this (is it "crypto snake oil"?)
  - The synchronization of the time interval for pairwise key negotiation is critical:
    - How do the nodes know when this starts? Should there be a signal?
    - What if a node misses this signal or "sleeps" during the interval?
    - If any node is compromised before erasure of K_I "all security is gone"...

TELEMATIK Rechnernetze

Alternative Approaches to Key Management (11)

- Remarks to some security aspects of LEAP (cont.):
  - What is the purpose of the nonce in the pairwise shared key establishment dialogue?
    - Pairwise shared keys are only established during T_{min}
    - Most probably, all neighbors will answer to the first message anyway (including the same nonce from this message...)
    - The nonce is not even included in the computation of $K_{u,v}$
    - The only thing that can be defended against with it, is an attacker that sends replayed replies during T_{min}, but these would not result in additional storage of keys $K_{u,v}$ or anything else than having to parse and discard these replays
  - The cluster key establishment protocol does not allow a node to check the authenticity of the received key, as every attacker could send some binary data that is decrypted to "something":
    - This would overwrite an existing cluster key K_u^c with garbage ($\Rightarrow$ DoS)
    - By appending a MAC this could be avoided (needs also additional replay protection in order to avoid overwriting with old keys)

Alternative Approaches to Key Management (12)

- Probabilistic key management:
  - Motivation:
    - Sharing one key K_G among all sensors leads to weak security
    - Sharing individual keys $K_{i,j}$ among all nodes *i, j* requires too many keys in large sensor networks ($n^2 - n$ keys for n nodes)
  - Idea [EG02]:
    - Randomly give each node a so-called "key ring" containing a relatively small number of keys from a large key pool
    - Let neighboring nodes discover the keys they share with each other
    - By properly adjusting the size of the key pool and the key rings, a "sufficient" degree of shared key connectivity for a given network size can be attained
  - The basic scheme published in [EG02] consists of three phases:
    - Key pre-distribution
    - Shared key discovery phase
    - Path key establishment phase

Alternative Approaches to Key Management (13)

- Key pre-distribution (5 offline steps):
  - Generate a large key pool P (~ 2^{17} - 2^{20} keys) with key identifiers
  - For each sensor randomly select *k* keys out of P without replacement in order to establish the sensor's key ring
  - Load every sensor with its key ring (= keys and their ids)
  - Load all sensor ids with the key ids of their key ring into a controller node
  - Load a shared key for secure communication with each sensor *s* into the controller node *ci*:
    - If $K_1, \ldots, K_k$ denote the keys on the key ring of sensor *s*, the shared key $K_{ci,s}$ is computed as: $K_{ci,s} = E(K_1 \oplus \ldots \oplus K_k, ci)$

Alternative Approaches to Key Management (14)

- The probability that two key rings KR1, KR2 share at least one common key can be computed as follows:
  - Pr(KR1 & KR2 share at least one key) = 1 - Pr(KR1 & KR2 share no key)
  - The number of possible key rings is: $\binom{P}{k} = \frac{P!}{k!(P-k)!}$
  - The number of possible key rings after k keys have been drawn from the key pool without replacement is: $\binom{P-k}{k} = \frac{(P-k)!}{k!(P-2k)!}$
  - Thus the probability that no key is shared is the ratio of the number of key rings without a match divided by the total number of key rings
  - Concluding the probability of at least one common key is:

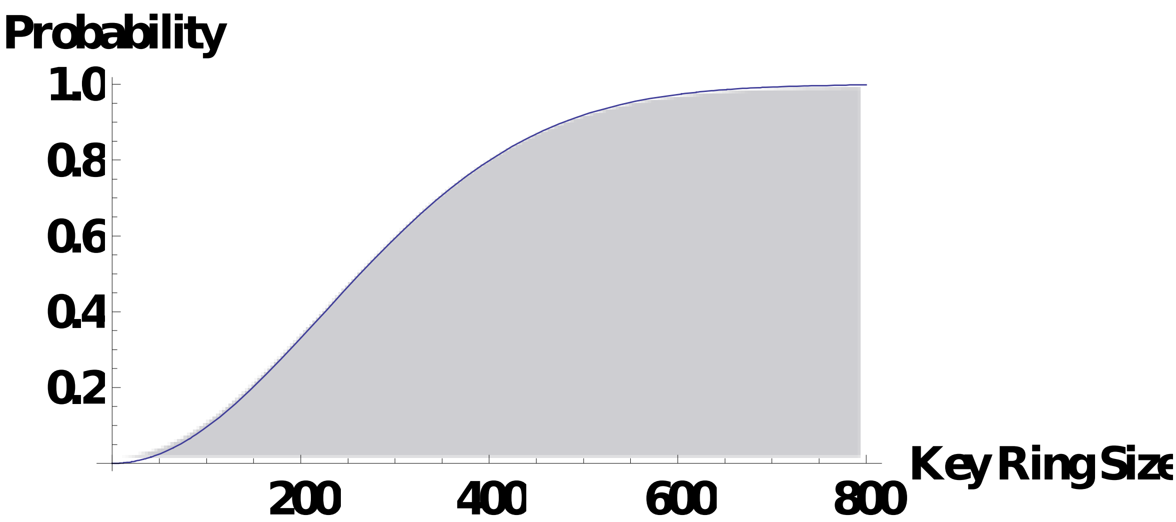
$$Pr(\geq 1 \text{ common key}) = 1 - \frac{k!(P-k)!(P-k)!}{P!k!(P-2k)!}$$

Alternative Approaches to Key Management (15)

- For how many links there is no key?

$$Pr(\geq 1 \text{ common key}) = 1 - \frac{(P-k)!^2}{P!(P-2k)!}$$

- Example for #Pool = 100 000



Alternative Approaches to Key Management (16)

- Shared key discovery phase:
  - After being installed, all sensor nodes start discovering their neighbors within the wireless communication range
  - Any two nodes wishing to find out if they share a key and simply exchange lists of key ids on their key ring
  - Alternatively, each node *s* could broadcast a list:
    - $s \rightarrow * : \alpha, E(K_1, \alpha), ..., E(K_k, \alpha)$
  - A node receiving such a list would then have to try all its keys in order to find out (with a high probability) matching keys
  - This would hide from an attacker which node holds which key ids
  - The shared key discovery establishes a (random graph) topology in which links exist between nodes that share at least one key
  - It might happen that one key is used by more than one pair of nodes

Alternative Approaches to Key Management (17)

- Path key establishment phase:
  - In this phase, path keys are assigned to pairs of nodes (s_1, s_n) that do not share a key but are connected by two or more links (so that there is a sequence of nodes which share keys and "connect" s_1 to s_n)
  - The article [EG02] does not contain any clear information on how path keys are computed / distributed:
    - It only states that they do not need to be generated by the sensor nodes
    - "The design of the DSN ensures that, after the shared key discovery phase is finished, a number of keys on any ring are left unassigned to any link"
    - However, it does not become clear from [EG02] how two nodes make use of these unused keys for establishing a path key

Alternative Approaches to Key Management (18)

- Node revocation:
  - If a node is detected to be compromised, all keys on its ring need to be revoked
  - For this, the controller node generates a signature key K_e and sends it individually to every sensor node *si*, encrypted with the key $K_{ci,si}$:
    - ci $\rightarrow$ si: $E(K_{ci,si}, K_e)$
  - Afterwards it broadcasts a signed list of all identifiers of keys that have to be revoked:
    - $s \rightarrow * : id_1, id_2, ..., id_k, MAC(K_e, id_1, id_2, ..., id_k)$
  - Every node receiving this list has to delete all listed keys from his key ring
  - This removes all links to the compromised node plus some more links from the random graph
  - Every node that had to remove some of its links tries to re-establish them by starting a shared key discovery and a path key establishment phase

Alternative Approaches to Key Management (19)

- Modifying the basic random pre-distribution scheme by requiring to combine multiple shared keys [CPS03]:
  - In this variant, two nodes are required to share at least *q* keys on their rings, in order to establish a link
  - If $K_1, ..., K_{q'}$ are the common keys of nodes *u* and *v* (with $q' \geq q$), then the link key is computed as follows: $K_{u,v} = h(K_1, ..., K_{q'})$
  - One the one hand side, it becomes harder with this scheme for an attacker to make use of a key ring(s) obtained by node compromise (increase is exponential in *q*)
  - On the other hand, the size of the key pool |P| has to be decreased in order to have a high enough probability that two nodes share enough keys on their rings in order to establish a link
    - This gives an attacker a higher percentage of compromised keys per compromised nodes ($\rightarrow$ tradeoff)
  - In [CPS03] a formula is derived how to compute the key pool size so that any two nodes share enough keys with probability > p
  - This scheme is called the *q-composite scheme*

Alternative Approaches to Key Management (20)

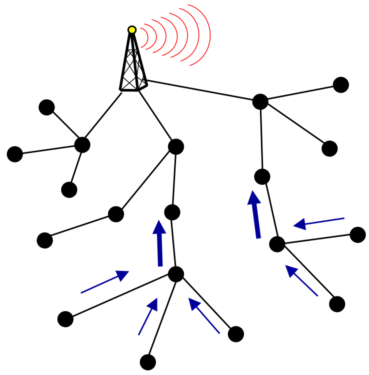
- Multipath key reinforcement:
  - Basic idea: "strengthen" an already established key by combining it with random numbers that are exchanged over alternative secure links
  - Assume that the discovery phase of the basic scheme has been completed and that enough routing information can be exchanged so that node *u* knows all (or enough) disjoint paths $p_1, \ldots, p_j$ to node *v*
  - Node *u* generates *j* random values $v_1, \ldots, v_j$ and sends each value along another path to node *v*
  - After having received all j values node v computes the new link key:
    - $K'_{u,v} = K_{u,v} \oplus v_1 \oplus \ldots \oplus v_j$
  - Clearly, the more paths are used, the harder it gets for an attacker to eavesdrop on all of them
  - However, the probability for an attacker to be able to eavesdrop on a path increases with the length of the path
  - In [CPS03] the special case of 2-hop multipath key reinforcement is analyzed probabilistically

Alternative Approaches to Key Management (21)

- Some security remarks on probabilistic key management:
  - The nice property of having a rather high probability that any two given nodes share at least one key (e.g. p = 0.5, if 75 keys out of 10,000 keys are given to every node), also plays in the hands of an attacker who compromised a node:
    - An attacker that has compromised more than one node has an even higher probability of holding at least one key with any given node
    - This problem also exists with the q-composite scheme (as the key pool size is reduced to ensure a high enough probability)
    - This especially concerns the attacker's ability to perform active attacks
    - Eavesdropping attacks are less probable as the probability that the attacker holds exactly the key that two other nodes are using is rather small (and even a lot smaller in the q-composite scheme)
  - Keys of compromised nodes are supposed to be revoked, but as how to detect compromised nodes still is an open question, how to know which nodes / keys to revoke?
  - The presented schemes do not support node-to-node authentication

Secure Data Aggregation (1)

- Remember that data from different sensors is supposed to be aggregated on its way towards the base station:
  - This raises the question, how to ensure integrity in this case?



  - If every sensor would add a MAC to its data in order to ensure data origin authentication, all (data, MAC)-tuples would have to be send to the base station

  ⇒ Individual MACs are not suitable for data aggregation!

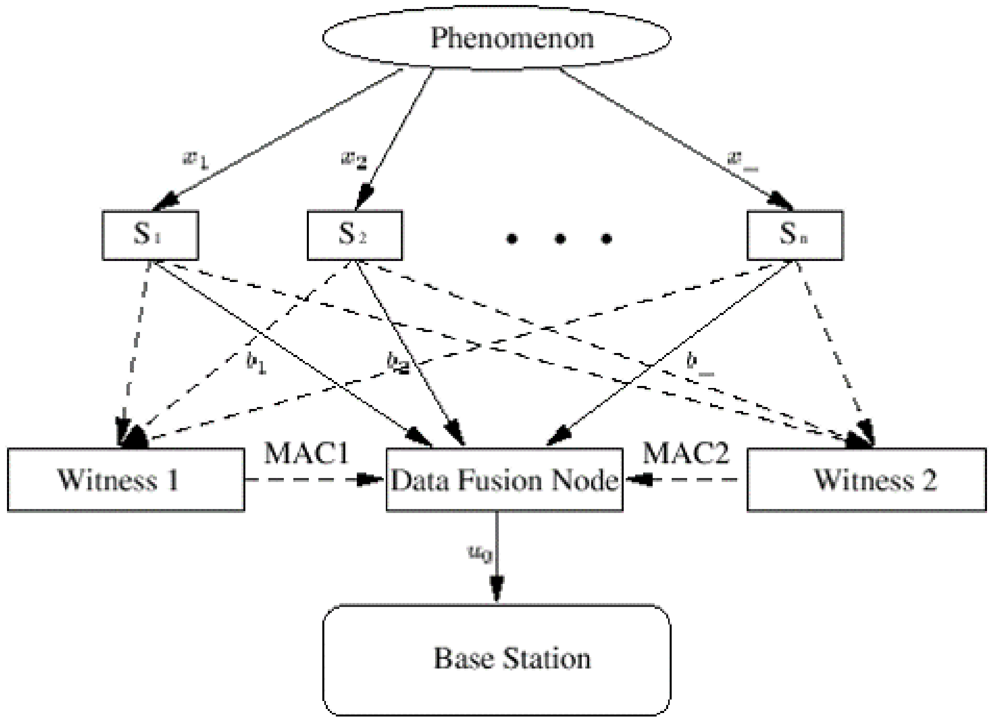
  - If only the aggregating node adds one MAC, a subverted node could send arbitrary data regardless of the data send by sensors

TELEMATIK Rechnernetze

Secure Data Aggregation (2)

- At *GlobeCom'03,* W. Du et. al. have proposed a scheme [DDH+03] that allows a base station to "check the integrity" of an aggregated value based on endorsements provided by so-called *witness nodes*:
  - Basic idea: multiple nodes perform data aggregation & "sign" their result
  - Requires individual keys between each node and the base station
  - In order to allow for aggregated sending of data, some nodes act as so-called *data fusion nodes*, aggregating sensor data and sending it towards the base station
  - As a data fusion node could be a subverted or malicious node, his result needs to be endorsed by witness nodes
  - For this, neighboring nodes receive the same sensor readings, compute their own aggregated result, compute a MAC over this result and send it to the data fusion node
  - The data fusion node computes a MAC over his own result and sends it together with all received MACs to the base station

Secure Data Aggregation (3)



(source: [DDH+03])

Secure Data Aggregation (4)

- Detailed scheme as described in [DDH+03]:
  - Sensor nodes $S_1, S_2, \ldots, S_n$ collect data from their environment and make their binary decisions $b_1, b_2, \ldots, b_n$ based on some detection rules
  - Every sensor node sends its decision to the data fusion node F
  - The data fusion node F computes an aggregated decision S_F
  - Neighboring witness nodes $w_1, w_2, \ldots, w_m$ also receive the sensor readings and compute their own fusion results $s_1, s_2, \ldots, s_m$
  - Every w_i computes a message authentication code with a shared key k_i it shares with the base station: $MAC_i = h(s_i, w_i, k_i)$
  - All w_i sends their MAC_i to the data fusion node
  - Variant A: $m+1$ out of $m+1$ voting scheme
    - F computes $MAC_F = h(S_F, F, k_F, MAC_1 \oplus MAC_2 \oplus \ldots \oplus MAC_m)$
    - F sends to base station: $(S_F, F, w_1, \ldots, w_m, MAC_F)$
    - Base station computes all $MAC'_i = h(S_F, w_i, k_i)$ and $MAC'_F = h(S_F, F, k_F, MAC_1 \oplus MAC_2 \oplus \ldots \oplus MAC_m)$
    - Base station checks if $MAC'_F = MAC_F$

Secure Data Aggregation (5)

- Some remarks on the *(m+1)* out of *(m+1)* scheme [DDH+03]:
  - If the set $(w_1, \ldots, w_m)$ remains unchanged, the identifiers of the w_i need only to be transmitted with the first MAC_F
  - However, if one witness deliberately sends a wrong MAC_i the aggregated data gets refused by the base station (⇒ risk of denial of service)
  - This calls for a less vulnerable alternative
- Variant B: *n* out of *m+1* voting scheme
  - *F* sends $(S_F, F, MAC_F, w_1, MAC_1, \ldots, w_m, MAC_m)$
  - The base station checks if at least *n* out of *m+1* MACs match, that is at least *n-1* MAC_i match MAC_F
  - This scheme is more robust against erroneous or malicious witness nodes, but requires a higher communication overhead as *m* MACs must be send to the base station

Secure Data Aggregation (6)

- In [DDH+03], Du et. al. analyze the minimum length of the MACs in order to ensure a certain tolerance probability $2^{-\delta}$ that an invalid result is accepted by a base station:
  - Assumptions:
    - Each MAC has the length *k*
    - There are *m* witnesses
    - No witness colludes with *F*
    - F needs to guess the endorsements MAC_i for at least *n-1* witnesses
  - As the probability of correctly guessing one MAC_i is $p=1/2^k$ the authors compute the chance of correctly guessing at least *n-1* values to:

$$P_s = \sum_{i=n-1}^{m} \binom{m}{i} p^i (1-p)^{m-i}$$

  - After some computation they yield: $m\left(\frac{k}{2} - 1\right) \geq \delta$

Secure Data Aggregation (7)

- Du et. al. conclude that it is sufficient if mk ≥ 2(δ+m)
- Example:
  - δ = 10 so that the probability of accepting an invalid result is 1/1024, and there are m = 4 witnesses ⇒ k ≥ 7
  - This observation is supposed to enable economizing transmission effort
- How to obtain a result if a data fusion node is corrupted?
  - In case that the verification at the base station fails, the base station is supposed to poll witness stations as data fusion nodes
  - [DDH+03] compute the expected number of polling messages T(m+1, n) to be transmitted before the base station receives a valid result:
    - Assumption: the probability of a node being compromised is p_c
    - With this, they obtain: $T(m+1, n) = p_c^{m-n+1} \sum_{K=n}^{m} p_c^{m-k} f(k, n)$

    with $f(m, n) = 1 + (m - n + 1)(1 - p_c)(1 - p_m^{n-1})$

    and p_j^i denoting the probability that out of *j* nodes at least *i* are honest

Secure Data Aggregation (8)

- Security discussion:
  - Let us think about for a moment, if an attacker actually needs to guess MACs in order to send an invalid result
  - As all messages are transmitted in clear, an eavesdropper E can easily obtain valid MACs: $MAC_i = h(s_i, w_i, k_i)$
  - If E later on wants to act as a bogus data fusion node sending an (at this time) incorrect result s_i he can replay MAC_i to support this value
  - As [DDH+03] assumes a binary decision result, an attacker only needs to eavesdrop until he has received enough MAC_i supporting either value of s_i
  - Thus, the scheme fails completely
- Could the scheme be “repaired”?
  - The reason for the vulnerability described above is the missing verification of the freshness of a MAC_i at the base station
  - A quick fix might be the base station regularly sending out random numbers r_B that have to be included in the MAC computations (every r_B is only accepted for one result, requiring large random numbers)
  - Alternative: time stamps, requiring synchronized clocks

Secure Data Aggregation (9)

- More remarks:
  - What happens if some witnesses can not receive enough readings?
  - Why are the MAC_i not send directly from the witnesses to the base station?
    - This would allow for a direct *n* out of *m+1* voting scheme
  - How to defend against an attacker flooding the network with “forged” MAC_i (“forged” meaning arbitrary garbage that looks like a MAC)?
    - This would allow an attacker to launch a DoS attack as an honest fusion node could not know which values to choose?
    - One more “hotfix”: a local MAC among neighbors to authenticate the MAC_i?
  - I still would not want to rely on this “improved scheme”...
- Some more general conclusions from this:
  - Optimization is one of the attacker’s best friends ;o)
  - In security, we often learn (more) from failures...

Summary

- Wireless sensor networks are an upcoming technology with a wide range of promising applications
- As in other networks, security is crucial for any serious application
- Prevalent security objectives in wireless sensor networks:
  - Confidentiality and integrity of data
  - Availability of sensor network services (threats: DoS, attacks on routing, ...)
  - Severe resource constraints (memory, time, energy) and an ”unfair” power balance makes attaining these objectives particularly challenging
- First approaches:
  - Approaches proposed for wireless adhoc networks which are based on asymmetric cryptography are considered to be too resource consuming
  - Basic considerations on protection against DoS and attacks on routing
  - SNEP and μTESLA for end-to-end security are one exemplary approach
  - Up to now there are only few works on how to design security functions suitable for the specific communication patterns in sensor networks (especially with respect to data aggregation)

References (1)

[CPS03] H. Chan, A. Perrig, D. Song. *Random Key Predistribution Schemes for Sensor Networks.* IEEE Symposium on Security and Privacy, May 2003.

[DDH+03] W. Du, J. Deng, Y. Han, P. Varshney. *A Witness-Based Approach for Data Fusion Assurance in Wireless Sensor Networks.* Proceedings of the IEEE 2003 Global Communications Conference (Globecom'2003), San Francisco, CA, December, 2003, pp. 1435-1439.

[EG02] L. Eschenauer, V. D. Gligor. *A Key Management Scheme for Distributed Sensor Networks.* CCS'02, Washington, DC, USA, November 2002.

[HPJ02a] Y. Hu, A. Perrig, D. Johnson. *Wormhole Detection in Wireless Ad Hoc Networks.* Technical Report TR01-384, Rice University, June 2002.

[KW03a] H. Karl, A. Willig. *A Short Survey of Wireless Sensor Networks.* TKN Technical Report Series, TKN-03-018, TU Berlin, 2003.

[KW03b] C. Karlof, D. Wagner. *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures.* AdHoc Networks Journal, Volume 1, Issues 2-3, pages 293-315, Elsevier, September 2003.

[KY98a] B.S. Kaliski, Y.L. Yin. *On the security of the RC5 encryption algorithm.* RSA Laboratories Technical Report, TR-602, Version 1.0, 1998.

[Men97a] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press Series on Discrete Mathematics and Its Applications, Hardcover, 816 pages, CRC Press, 1997.

References (2)

[NSS+04] J. Newsome, E. Shi, D. Song, A. Perrig. *The Sybil Attack in Sensor Networks: Analysis & Defenses.* IPSN'04 , Berkeley, California, USA, April 2004.

[PS+02a] A. Perrig, R. Szewcyk, J. Tygar, V. Wen, D. Culler. *SPINS: Security Protocols for Sensor Networks.* Wireless Networks, vol. 8, pp. 521-534, Kluwer, 2002.

[W01a] A. Wood. *Security in Sensor Networks.* Sensor Networks Seminar, University of Virginia, Fall 2001.

[WS02a] A. Wood, J. Stankovic, *Denial of Service in Sensor Networks.* IEEE Computer, vol. 35, pp.54-62, October 2002.

[ZSJ03] S. Zhu, S. Setia, S. Jajodia. *LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks.* CCS'03, Washington, DC, USA , October 2003.