DISH: DIstributed Self-Healing
in Unattended Wireless Sensor Networks

Di Ma and Gene Tsudik

Computer Science Department
University of California, Irvine

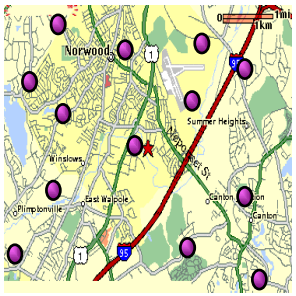
SSS 2008, Detroit

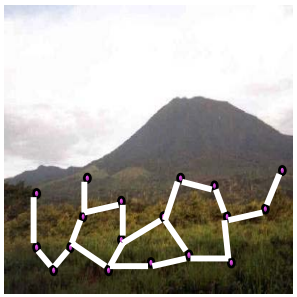
Outline

- Introduction to UWSNs
- Motivation
- DISH
- Conclusion

"Typical" Wireless Sensor Network



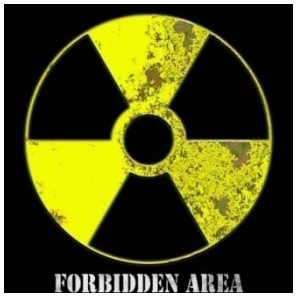




Many real, alleged and imagined applications

- Networking
  - Sensor-to-sink communication (opt. sink-to-sensors)
- Low cost sensors
  - Sensing environmental measurements
    - Periodically, event driven, or query-based (on-demand)
- Online sink
  - Real-time off-loading of data

Unattended Wireless Sensor Network (UWSN)

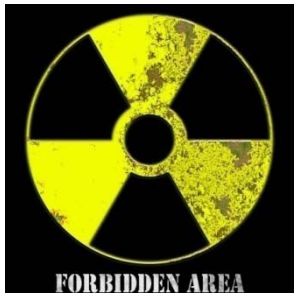






- UWSNs operate in **hostile** environments
  - Initial deployment might be ad-hoc
- No ever-present sink
  - Itinerant, visits UWSN at irregular intervals
- Nodes are mostly left on their own
  - Nodes might retain data for a long time
  - Data might be valuable
- Advantages to the adversary
  - Adversary roams around with impunity
  - Adversary has **lots of time**

Examples of UWSNs



- WSN in a recalcitrant country monitoring nuclear activity



- Underground WSN monitoring sound and vibration produced by troop movements or border crossings



- Anti-poaching WSN in a national park tracking/recording firearm discharge locations

New Adversary Model (ADV)

- Previous adversaries corrupt fixed number of nodes over all time (k out of n)
  - Solutions focused on detection
  - Once attack is detected, on-line sink can help
    - e.g., exclude compromised nodes
- Our adversary is MOBILE
  - Roams the network, compromises k-out-of-n nodes per round
    - Given enough time, can subvert the entire UWSN
  - Sink is off-line: real-time detection does not help
    - ADV can come and go with impunity

Motivation

- Curious adversary wants to read/learn sensor-collected data
- Encryption does not help
  - Symmetric keys are exposed with node compromise
  - With public key encryption, ADV can GUESS the plaintext
    - Randomized public key encryption helps but only with a TRNG
    - Not currently available (nor foreseeable)
- Can we protect category (1) and (3) data?

Sensor data can be partitioned, based on time of compromise:
(1) Collected before compromise
Need forward-secure encryption
(2) Collected during compromise
Nothing helps…
(3) Collected after compromise
Need backward-secure encryption

Forward Secrecy

- Even if ADV learns the current key, it cannot derive PREVIOUS round keys
- Per-round key evolution
  - At the end of round r, next round key computed using one-way function (current key is securely erased)

$$K^{r+1}=H(K^r)$$

- But, after compromise, ADV can mimic key evolution process



$K^1 \rightarrow K^2 \rightarrow K^3 \rightarrow K^4 \rightarrow K^5 \rightarrow K^6 \rightarrow K^7 \rightarrow \ldots$

Sensor compromised at round 4 and then released



$\leftarrow K^4 \rightarrow K^5 \rightarrow K^6 \rightarrow K^7 \rightarrow \ldots$

Backward Secrecy

- Even if ADV learns the current key, it cannot derive FUTURE round keys
- Also, based on per-round key evolution
  - Requires online TTP, or secure hw at each node (same as distributed TTP)
- Not suitable for UWSNs
  - Our sink is off-line
  - A single sensor cannot act as a TTP for peers since it can be compromised at any time



$K^1 \rightarrow K^2 \rightarrow K^3 \rightarrow K^4 \rightarrow K^5 \rightarrow K^6 \rightarrow K^7 \rightarrow \ldots$



$K^1 \leftarrow K^2 \leftarrow K^3 \leftarrow K^4 \rightarrow$ ⛔

Sensor compromised at round 4 and then released

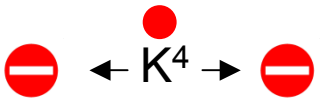
Key Insulated Secrecy

- Schemes that are both BACKWARD and FORWARD secure are known as KEY INSULATED schemes
  - Public key based
  - Require online trusted third party to update private keys
  - Against compromise of sink's private key
    - Not applicable for UWSNs



$K^1 \rightarrow K^2 \rightarrow K^3 \rightarrow K^4 \rightarrow K^5 \rightarrow K^6 \rightarrow K^7 \rightarrow \ldots$

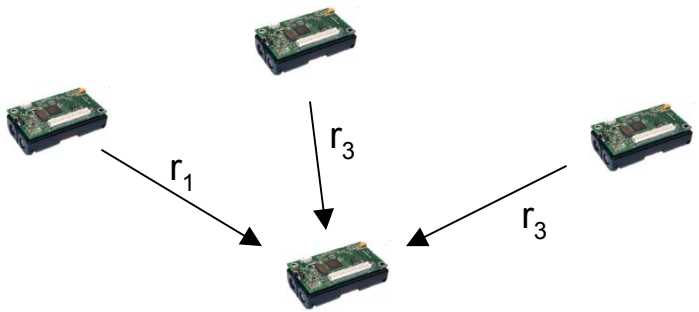




Sensor compromised at round 4 and then released

DISH: Main Idea

- Forward secrecy through key evolution
- Backward secrecy through sensor cooperation
  - A sensor can securely generate a key unknown to ADV, if it obtains at least one contribution from a non-compromised peer



$K_{new} = F(K_{old}, r_1, r_2, r_3)$

Network Model 1/2

- **Periodic data collection**
  - Time divided in equal and fixed collection rounds
  - Each sensor collects a single data unit per round
- **Storage**
  - ***v*** – maximum number of collection rounds between successive sink visits.
  - Each sensor has enough storage for O(v) data
- **Communication**
  - UWSN is always connected
  - Any two sensors can communicate either directly or through peers

Network Model 2/2

- **Cryptographic Capabilities**
  - Cryptographic hashing, e.g., SHA-2
  - Symmetric key encryption
    - unique secret key shared with the sink
  - Pseudo-Random Number Generator (PRNG)
    - unique secret seed shared with the sink
- **Unattended Operation**
  - Itinerant sink periodically visits to collect data.
- **Re-initialization**
  - During each visit, sink re-initializes ALL sensors (ADV not present):
    - New secret key
    - New secret seed
    - Empty storage

Adversarial Model

- **Goal**
  - Learn as many secrets as possible
- **ADV compromise power**
  - Compromises at most $0 < k < n/2$ nodes per round
  - Reads state/storage
  - Listens to all incoming/outgoing communication
- **Minimal disruption**
  - ADV does not interfere with sensor behavior (e.g., in order to stay undetected)
- **Periodic operation**
  - At the end of each round, picks a subset of sensors up to k
  - At the start of each round, atomically releases current sensors and compromises the new subset
- **Topology Knowledge**
  - Knows the entire topology

DISH: main process

Algorithm 1: DISH: sensor s_i main process at round r

/* **start round r** */
1 collect new sensed data d_i^r
2 compute $E_i^r = Enc(K_i^r, d_i^r, r, \cdots)$ ← Normal sensor operations
3 store E_i^r on local storage
4 set $HELP[1..t] = \emptyset$
5 **for** $(p \leftarrow 1$ **to** $t)$ **do**
5.1 set $j = PRNG()$ mod n ← Pick sponsors
5.2 send $REQUEST$ to s_j
5.3 set $HELP[p] = REPLY(s_j)$
6 set $K_i^{r+1} = \mathscr{H}(K_i^r, HELP[1], ..., HELP[t])$ ← Key refresh
7 erase K_i^r and $\{HELP[1], ..., HELP[t]\}$
/* **end round r** */

DISH: helper process

Algorithm 2: DISH: sensor s_i helper process at round r

```
   /* start round r */
   while (;;) DO do
1      receive REQUEST from s_j
2      set HELP = PRNG()
3      compose REPLY using HELP
4      send REPLY to s_j
5      erase HELP
   /* end round r */
```

Sensor States



Occupied sensors (R^r)

- currently controlled by ADV
- k per round

Sick sensors (Y^r)

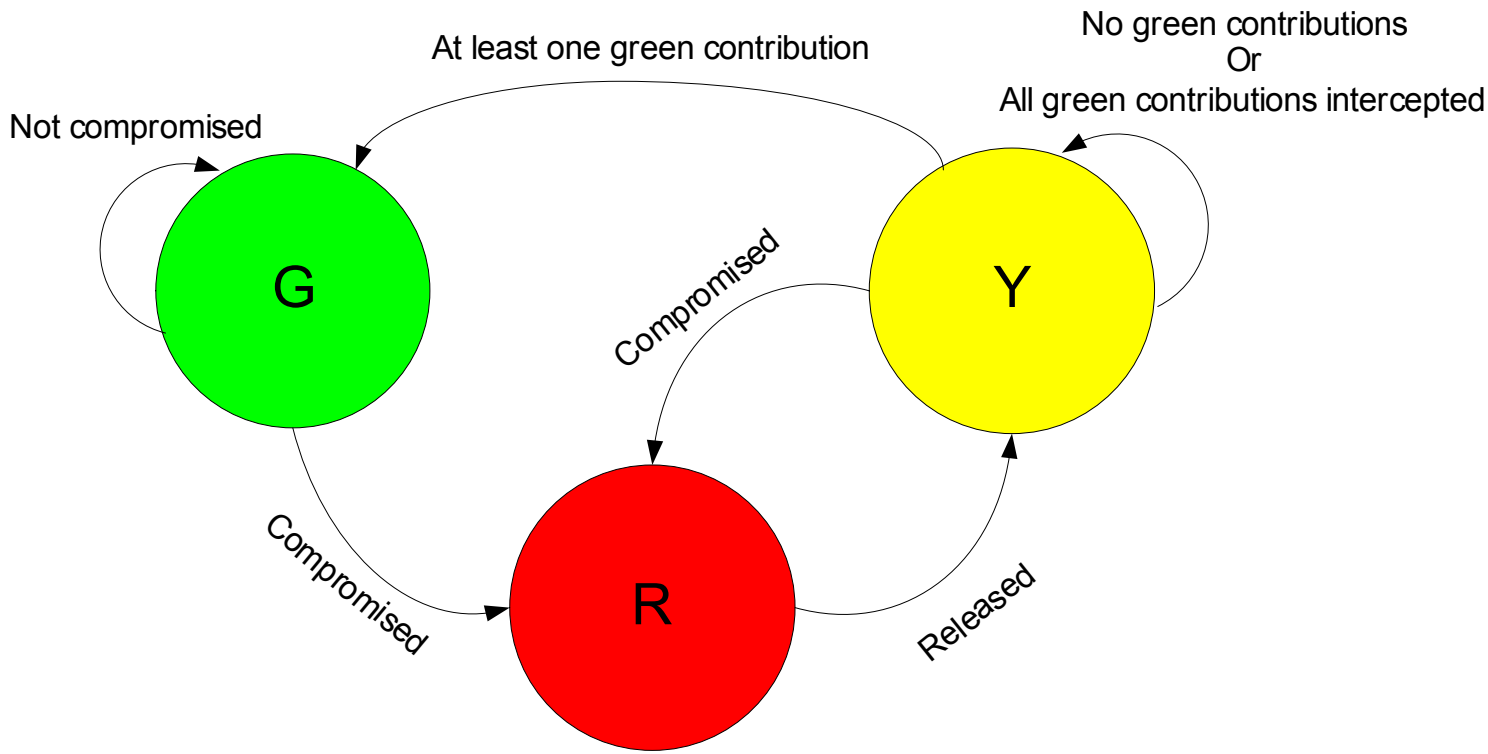
- Compromised in some previous round
- Their current keys are known to ADV



Healthy sensors (G^r)

- Either never been compromised
- **Or** recovered through DISH

Sensor State Transition Diagram

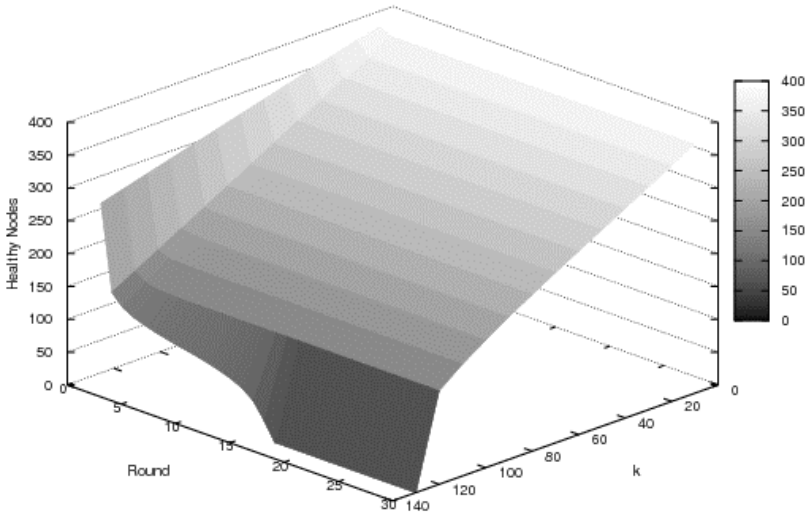


- |R|=k
- ADV's goal it to maximize |Y|+|R|
- Network goal: max(|G|)=n-2k

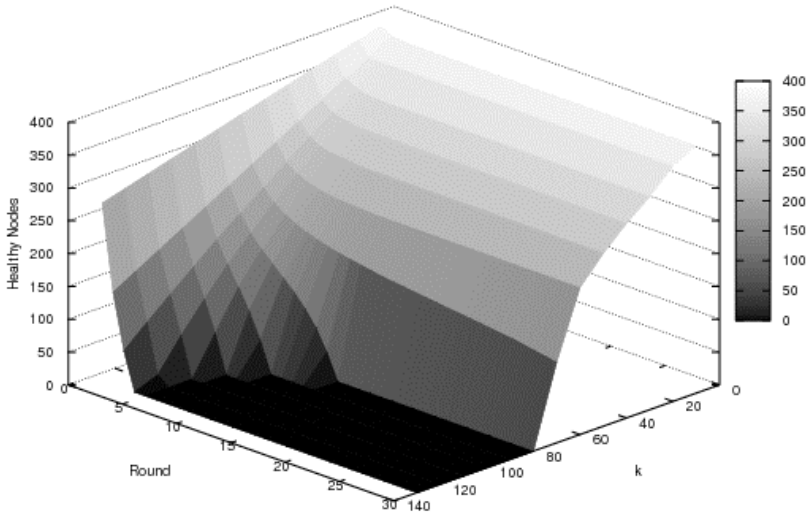
Analytical Setting

- ADV type: random
  - An adversary which compromises a new set of **healthy** sensors each round randomly
- Performance metric:
  - number of healthy sensors |G| each round
- Performance factors:
  - k = ADV compromise power
  - p = ADV eavesdropping capability
  - t = number of sponsors

Effect of k and p (power of ADV)



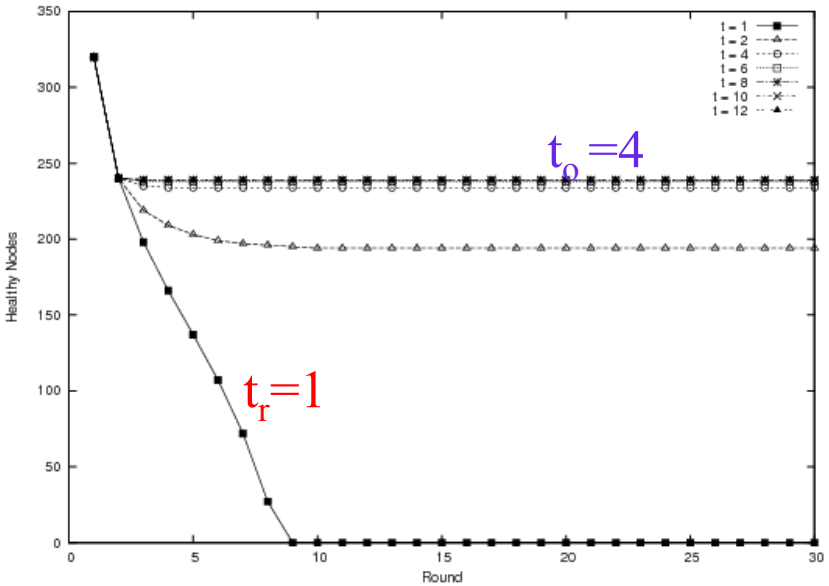
(a) $n = 400\ t = 6\ p = 0.2$



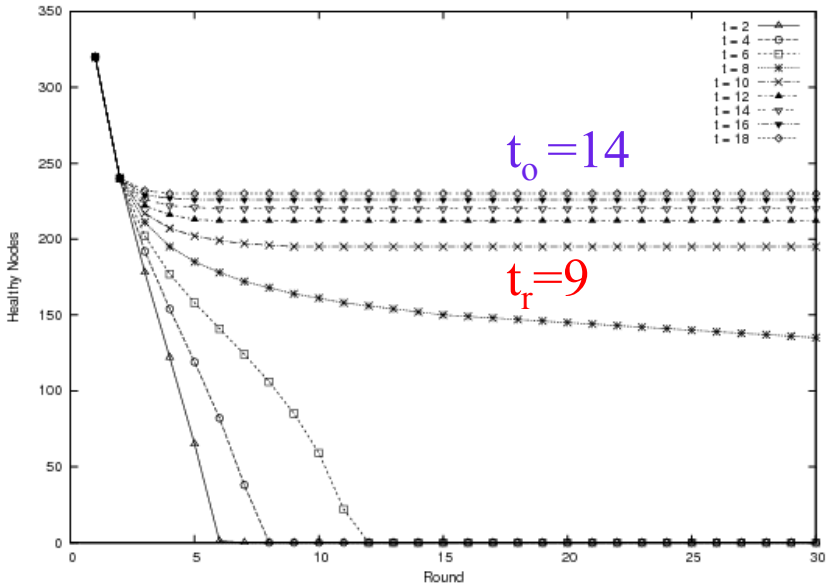
(b) $n = 400\ t = 6\ p = 0.8$

- Threshold phenomena:
  - for p=0.2, |G| remains stable for k/n < 127/400
  - for p=0.8, |G| remains stable for k/n < 68/400

Effect of t



(a) $n = 400$ $k = 80$ $p = 0.2$



(b) $n = 400$ $k = 80$ $p = 0.8$

Two threshold values for t

- t_r: determines whether the network can win
- t_o: determines the communication overhead

Dealing with the "Real World"

- Communication Failure
  - Sink synchronization
  - Store the IDs of missed sponsors
- Sensor Failure
  - If a sensor dies, key evolution history (of other sensors) cannot be reconstructed
  - Other sensors might depend on the failed node
- Public Key encryption
  - Encrypt round key under the sink PK and use round key for everything else
  - Use the sink PK to encrypt data and use round key as random number

Summary

- UWSN is a new, exciting field that calls for innovative security solutions
- Sensor cooperation is essential to achieve security against a powerful mobile adversary
- More advanced ADV types need to be treated

Questions?

Thank you!