SPINS: Security Protocols for Sensor Networks

Adrian Perrig, Robert Szewczyk, J.D. Tygar, Victor Wen, and David Culler
Department of Electrical Engineering & Computer Sciences, University of California - Berkeley

Introduction

- Wireless sensor networks are increasingly prevalent (or at least that was the prevailing thought in 2002)
- Sensors are very resource-limited (SmartDust)
  - Slow communication links (10 kbps)
  - Limited computing power (8-bit, 4 MHz)
  - Limited memory and storage (512 bytes)
  - Limited battery life
  - TinyOS

Sensor Applications

- Emergency response
  - Buildings, roads, airports, etc.
- Energy management
  - Mitigate blackouts by sensing temperature and load balance information and redistributing power
- Medical monitoring
  - Automatic medication administration
- Inventory management
  - Distribution tracking
- Battlefield management
  - Collect and distribute information about battlefield conditions

Motivation

- Some of these applications are critical
- Security is often ignored
  - Too much power
  - Too much communication overhead
  - In some cases, not enough memory to even store the parameters!
    - 1024-bit RSA

Is security on sensors possible?

- Remember, the devices are very resource-constrained
- Asymmetric cryptography in particular is both:
  - Computationally intensive, which shortens battery life
  - Overhead intensive, which decreases overall efficiency AND shortens battery life
    - In wireless sensors, communications make up the majority of energy consumption
    - Overhead can be as long as 1000 bytes per packet!
- TESLA, a protocol developed for authenticated broadcast, is unsuitable for sensors

TESLA

- Broadcast authentication mechanism using only symmetric cryptographic primitives
- Receivers should be able to verify authentication data but not generate it
- Senders and receivers should be loosely time-synchronized
- Senders use one-way key chaining (more on this later)
- Receivers only accept packets generated with secret keys

The SPINS Approach

- Two components:
  - SNEP (Sensor Network Encryption Protocol)
    - Provides cryptographic strength, two-party data authentication, replay protection, freshness, and integrity
  - μTESLA
    - Provides broadcast authentication
- Each station has a shared secret key with the base station
- All cryptographic operations based on a single block cipher

Architecture Assumptions

- Sensor networks have one or more base stations
- Base stations have significantly more power
- Periodic beacons establish routing topology
- Individual nodes communicate through the base station
- Three types of communication:
  - Node to base station
  - Base station to node
  - Broadcast (from base station)

Trust Assumptions

- Individual nodes are not trusted, but they do trust themselves (at least in terms of synchronization)
- The base station is trusted
- Broadcast medium is not trusted
- Single-node compromise should not compromise the rest of the network

Security Requirements

- Confidentiality
  - Transmissions should be recognizable only by authorized receivers
- Authentication
  - All messages must be verified as coming from trusted sources
- Integrity
  - Data is not altered in transit
- Freshness
  - Data is not outdated
- A ***secure channel*** combines all of the above

The Building Blocks

- Sensor Network Encryption Protocol (SNEP)
  - Provides data confidentiality
  - Authentication
  - Integrity
  - Freshness
  - *Semantic security*
    - Identical messages encrypted differently using CTR mode

SNEP Counters

- Senders and receivers share counters
- One shared counter per direction
- Requires synchronization
- Counter exchange (resynchronization) is possible

Message Authentication

- Each pair of entities shares a master key X_{AB}
- Pseudorandom functions allow for four keys to be generated from this master key
  - K_{AB} – Encryption from A to B
  - K_{BA} – Encryption from B to A
  - K'_{AB} – MAC from A to B
  - K'_{BA} – MAC from B to A
- Using different keys for encryption and MAC reduces weaknesses from potential interaction

Data Transmission

$$A \rightarrow B: \quad \{D\}_{\langle K_{AB}, C_A \rangle}, \ \mathrm{MAC}\left(K'_{AB} C_A \,||\, \{D\}_{\langle K_{AB}, C_A \rangle}\right). \quad (1)$$

- Remember the shared counter
- Communication overhead is low since the counter is not transmitted (8 bytes)
- Counter enforces an ordering of messages (weak freshness)
- For strong freshness, send a request message (R) with a *nonce*

$$\begin{aligned} &A \rightarrow B: \quad N_A, R_A, && (2) \\ &B \rightarrow A: \\ &\quad \{R_B\}_{\langle K_{BA}, C_B \rangle}, \ \mathrm{MAC}\left(K'_{BA}, N_A \,||\, C_B \,||\, \{R_B\}_{\langle K_{BA}, C_B \rangle}\right). \end{aligned}$$

Counter Exchange

$$\begin{array}{ll} A \rightarrow B: & C_A, \\ B \rightarrow A: & C_B,\ \mathsf{MAC}(K'_{BA} C_A \,||\, C_B), \\ A \rightarrow B: & \mathsf{MAC}(K'_{AB}, C_A \,||\, C_B). \end{array}$$

- Initial exchange does not require encryption
- Strong freshness is achieved by using counter values as nonces
- Resynchronization is a simple request/response pair

$$\begin{array}{ll} A \rightarrow B: & N_A, \\ B \rightarrow A: & C_B,\ \mathsf{MAC}(K'_{BA}, N_A \,||\, C_B). \end{array}$$

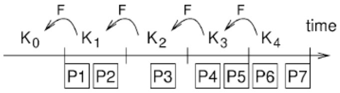
µTESLA: Authenticated Broadcast

- Very similar to TESLA, but with some changes to reduce overhead (standard TESLA is 24 bytes)
  - Sensor packets are only ~ 30 bytes
- No digital signatures are used to initially authenticate
  - Only symmetric mechanisms used
- Key disclosure is less frequent (once per time period instead of once per packet)
- The number of authenticated senders is restricted
- (Same problems as TESLA!)

μTESLA, continued

- Requires that communicating nodes are loosely time synchronized
- Also requires that each node knows the maximum synchronization error
- Time is divided into epochs, with one key used per epoch

μTESLA One-Way Key Chain



- Using a one-way function (such as MD5), some key K(j) can be generated as MD5(K(j+1))
- Keys are generated in reverse order (preventing the discovery of keys not yet known outside of the sender)
- Receivers buffer packets until keys are disclosed and the contents authenticated
  - When key K2 is disclosed, receivers can authenticate packets P1 and P2

μTESLA: Key Disclosure

- Keys are disclosed when:
  - Some time longer than any reasonable round-trip delay between the sender and receiver has passed
  - This prevents artificial packet injection since packets generated with a previously-disclosed key will be known to be outdated (and likely forged)

μTESLA: Adding Receivers

- New receivers need only one authentic key
  - The one-way chain allows verification of future keys
- Receivers must be loosely synchronized
- This requires strong freshness and two-party authentication
  - SNEP’s request/response pair works
  - Sender responds to a request with its current time, some key of the chain, the start time of a time interval, the duration, and the disclosure delay
  - Does not need to be encrypted

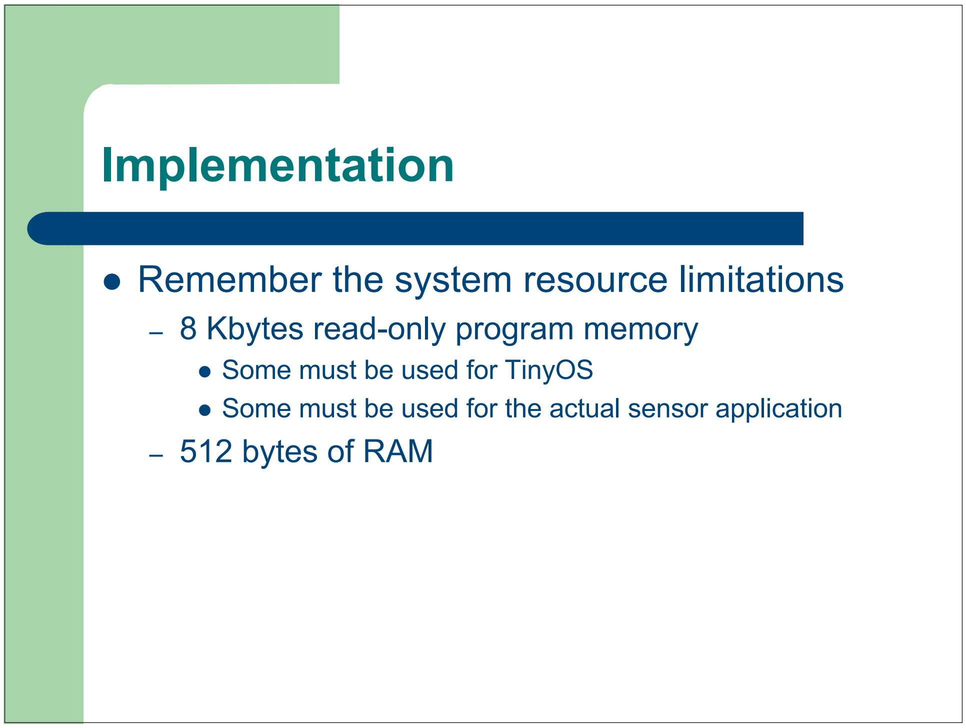
$$\begin{aligned} M \rightarrow S: &\quad N_M \\ S \rightarrow M: &\quad T_S \mid K_i \mid T_i \mid T_{\text{int}} \mid \delta \\ &\quad \text{MAC}(K_{MS}, N_M \mid T_S \mid K_i \mid T_i \mid T_{\text{int}} \mid \delta). \end{aligned}$$

µTESLA: Authenticating Packets

- Receivers discard packets that have unusually long delay
  - Could have been generated with already-disclosed keys
- Receivers can only verify packets once keys have been disclosed
- There is some inherent delay in authenticated broadcast since receivers must some time intervals before authenticating a received broadcast packet

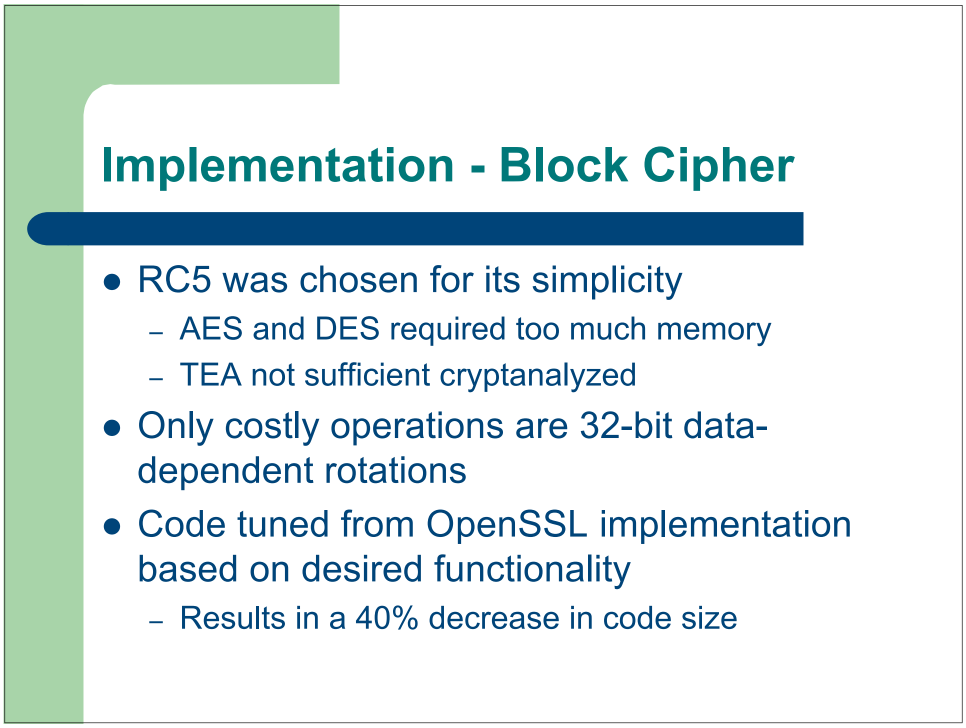
µTESLA: Node Broadcast

- Node memory is insufficient for one-way key chains, so nodes can either:
  - Broadcast through the base station using SNEP
  - Broadcasts data, but the base station handles the key chain (sending current values to the broadcasting node)
    - Generally too energy-intensive for a node, so the base station might disclose keys or handle adding receivers



Implementation

- Remember the system resource limitations
  - 8 Kbytes read-only program memory
    - Some must be used for TinyOS
    - Some must be used for the actual sensor application
  - 512 bytes of RAM



Implementation - Block Cipher

- RC5 was chosen for its simplicity
  - AES and DES required too much memory
  - TEA not sufficient cryptanalyzed
- Only costly operations are 32-bit data-dependent rotations
- Code tuned from OpenSSL implementation based on desired functionality
  - Results in a 40% decrease in code size

RC5 Operation & RNG

- CTR mode encryption used
  - Removes the need for separate decryption
  - Single-block error propagation good for wireless transmission environments
  - Enforces message ordering
- MAC function used to generate random numbers with

Message Authentication

- CBC-MAC is used
- One MAC computed per packet
  - Achieves both integrity and authentication since the MAC keys are unidirectional

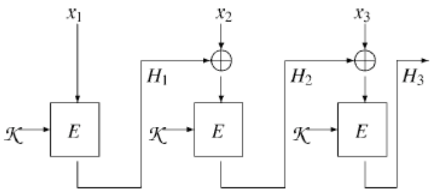
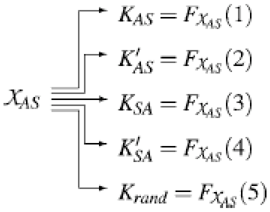


Figure 3. CBC MAC. The output of the last stage serves as the authentication code.

Key Derivation

- MAC function used to generate keys from the known shared master key (between each node and the base station)
  - $F_K(x)$ = MAC(K,x)
  - Each key is computationally independent



$$\begin{aligned} K_{AS} &= F_{\chi_{AS}}(1) \\ K'_{AS} &= F_{\chi_{AS}}(2) \\ K_{SA} &= F_{\chi_{AS}}(3) \\ K'_{SA} &= F_{\chi_{AS}}(4) \\ K_{rand} &= F_{\chi_{AS}}(5) \end{aligned}$$

Evaluation

- Differences arise from the implementation of the data-dependent rotation (a 32-bit operation) on an 8-bit processor
- Protocol itself is 574 bytes, for just over 2 Kbytes total

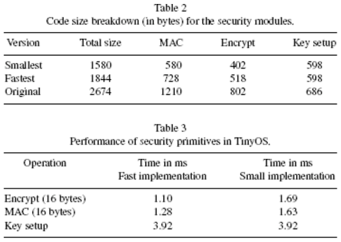


Table 2
Code size breakdown (in bytes) for the security modules.

Version	Total size	MAC	Encrypt	Key setup
Smallest	1580	580	402	598
Fastest	1844	728	518	598
Original	2674	1210	802	686

Table 3
Performance of security primitives in TinyOS.

Operation	Time in ms Fast implementation	Time in ms Small implementation
Encrypt (16 bytes)	1.10	1.69
MAC (16 bytes)	1.28	1.63
Key setup	3.92	3.92

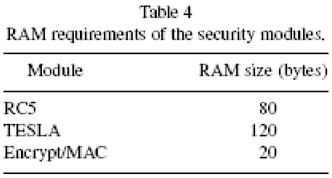


Table 4
RAM requirements of the security modules.

Module	RAM size (bytes)
RC5	80
TESLA	120
Encrypt/MAC	20

Aside: “Fast” is a relative term

- At 1.10 ms per encryption, total encryption throughput (without MAC) is about 116.4 kbps
- “Fast” FPGA-based encryption of AES can achieve a throughput of at least 30 Gbps
  - Only about 250,000 times faster

More Evaluation

- Key disclosure interval is 2
- To check validity, this means two key setup operations and two encryptions
- To check message integrity, two key setup operations, two encryptions, and up to four MAC operations are needed – 17.74 ms total
- Limiting factor is actually the amount of memory dedicated to buffering

Energy Usage (SNEP)

- Costs based on 30-byte packets
- Notice the costs are heavily skewed towards communication
- No additional cost for encrypted data transmission since encrypted block size is the same as plaintext

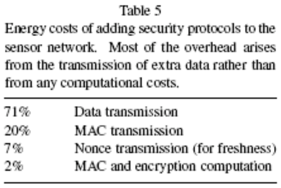


Table 5

Energy costs of adding security protocols to the sensor network. Most of the overhead arises from the transmission of extra data rather than from any computational costs.

71%	Data transmission
20%	MAC transmission
7%	Nonce transmission (for freshness)
2%	MAC and encryption computation

Energy Usage (μTESLA)

- Same as SNEP, but:
  - Periodic key disclosure combined with routing updates
  - Can be viewed as free if routing updates are considered necessary
  - Can also be viewed as wasted energy if authenticated routing is considered a waste

Other Security Issues

- No consideration of covert channels
- No consideration of compromised nodes
- No consideration of DoS attacks
- No consideration of non-repudiation

Applications

- Authenticated routing built on µTESLA
- Routing beacons broadcast periodically
- When nodes receive beacons, if they have not received a beacon in the current time interval they:
  - Accept the sender as a parent
  - Broadcast a routing beacon with itself as the sender
- µTESLA key disclosure packets can serve as beacons
  - Authenticity and freshness guaranteed
  - Nodes use watchdog behavior for anomaly detection (misbehaving nodes)

More Applications

- Node-to-node key agreement
- SNEP ensures strong freshness, confidentiality, and authentication using symmetric cryptography
- Nodes A and B use a mutually trusted base station S for exchange
- Base station does most of the work

$$\begin{array}{ll}
A \rightarrow B: & N_A, A, \\
B \rightarrow S: & N_A, N_B, A, B, \mathrm{MAC}\left(K'_{BS}, N_A|N_B|A|B\right), \\
S \rightarrow A: & \{\mathcal{SK}_{AB}\}_{K_{SA}}, \mathrm{MAC}\left(K'_{SA}, N_A|B|\{\mathcal{SK}_{AB}\}_{K_{SA}}\right), \\
S \rightarrow B: & \{\mathcal{SK}_{AB}\}_{K_{SB}}, \mathrm{MAC}\left(K'_{SB}, N_A|B|\{\mathcal{SK}_{AB}\}_{K_{SB}}\right).
\end{array}$$

Related Work

- Key distribution and key agreement in resource-constrained environments
- Asymmetric cryptography in ad-hoc networks
- Ad-hoc peer-to-peer authentication based on public key certificates
- Cryptography in relatively primitive devices

- (The paper has 57 references, 26 of which are cited in the related work section.)

Conclusion

- SNEP and μTESLA together provide secure communication channels using only symmetric cryptography in sensor networks
  - Confidentiality
  - Authentication
  - Integrity
  - Freshness
  - Low overhead

Contributions & Merits

- The SPINS method is a comprehensive security protocol for sensor networks using only symmetric cryptography
  - Relatively low communication overhead
  - Compact (runs on SmartDust)
  - Relatively resistant to compromise
- Pretty advanced for 2002

Contributions & Merits

- Combines two unique methods
  - SNEP & μTESLA
- Actually implemented on SmartDust sensors
  - Gives actual performance numbers on extremely resource-constrained environments
  - Some limited analysis on energy consumption
- Simple yet effective design choices
  - Use of a single block cipher for all operations
  - Counter mode encryption

Contributions & Merits

- SPINS is relatively universal and extensible to many other embedded applications
- Two application examples given
  - Authenticated routing in ad-hoc networks using key disclosure packets as routing beacons
  - Secure node-to-node key agreement using symmetric cryptography

Weaknesses & Drawbacks

- Weak mobility model
  - Sensor networks assumed to have a base station
    - What if they don't?
    - Lots of other papers assume nodes take turns being the base station, negating the "supernode" assumption
  - It appears mobility is limited or infrequent
    - If it isn't, the overhead from the routing beacons might be significant

Weaknesses & Drawbacks

- Time synchronization is a key assumption
  - Clock drift is actually a major problem in sensor networks using crystal oscillators
    - D. Scott, ACM SE Regional Conference, 2005
  - Packet loss is also potentially a major issue in wireless environments
- Both can be mitigated by resynchronizing the counter or sending it with the message
  - But this leads to huge (and potentially devastating) overhead in sensor networks!
- Clock drift could lead to attacks

Weaknesses & Drawbacks

- Only one cipher is used (RC5)
  - RC5 is simple, but does have weaknesses
- Other assumptions are inaccurate
  - AES doesn’t require lookup tables
  - TEA was cryptanalyzed (and broken) in 1997
  - XTEA and XXTEA existed (and were better options)
    - Extremely small code size (smaller than RC5)

Weaknesses & Drawbacks

- No non-repudiation
- No study of compromised nodes
- No study of the effects of error rates on energy consumption

Future Work & Extensions

- Consider testing other ciphers
- Given a more advanced platform (as we would expect with time) what can be done?
  - NTRU and Rabin for asymmetry
  - AES or RC6 for symmetry
- Test the effects of clock drift
- Test the effects of errors in transmission
- **Questions?**