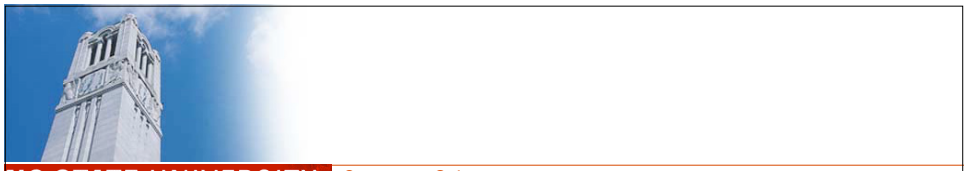


NC STATE UNIVERSITY Computer Science

CSC 774 Advanced Network Security

Topic 4.3 Mitigating DoS Attacks
against Broadcast Authentication in
Wireless Sensor Networks

1

Wireless Sensor Networks (WSN)

- A WSN consists of a potentially large number of battery powered sensor nodes
- Each node has sensors, actuators, micro controller, memory, radio
- Example sensor platforms
  - Low end: MICA series of motes
  - High end: Intel iMotes

MICA2

MICA2DOT

MICAz

Intel iMote

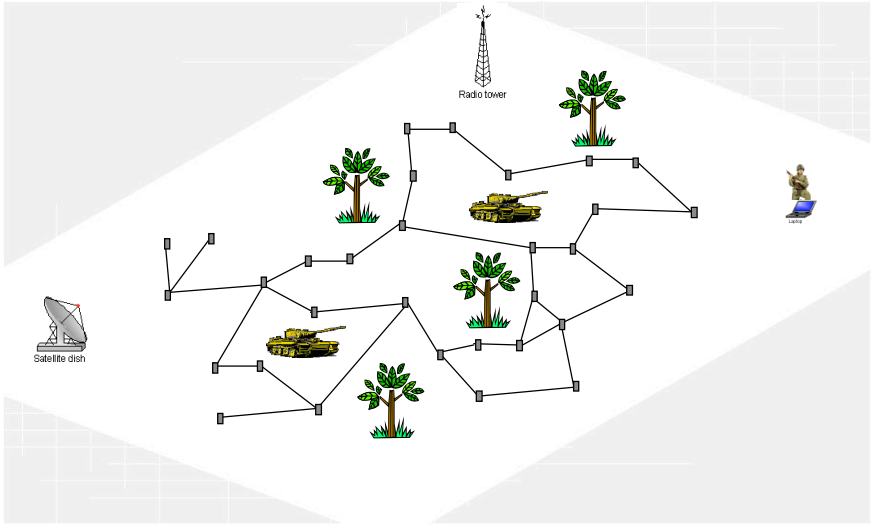
NC STATE UNIVERSITY

2

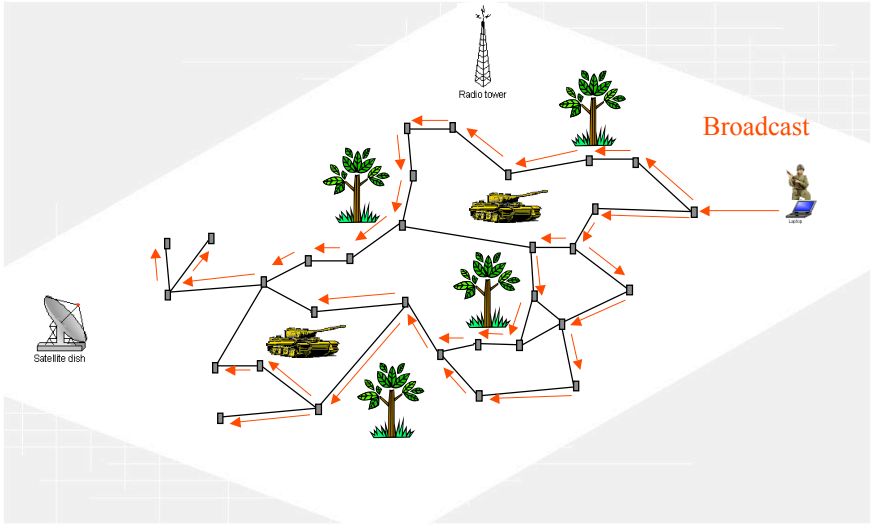
Applications of Wireless Sensor Networks

- Structural health monitoring
  - Buildings, bridges, etc.
- Target tracking
- Environmental monitoring
- Industry monitoring
- Battlefield surveillance

Wireless Sensor Networks (Cont'd)



Broadcast: A Critical Communication Primitive

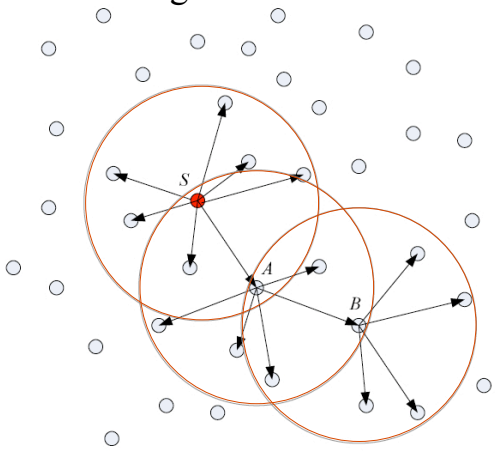


Broadcast Authentication

- In hostile environments, broadcast messages must be authenticated
- Options for broadcast authentication
  - Digital signature
    - Expensive
    - Possible for high-end sensor nodes
  - μTESLA and its variations
    - Based on symmetric cryptography ⇒efficient
    - Requires loose time synchronization
  - Threat: DoS attacks

Broadcast in Multi-Hop Sensor Networks

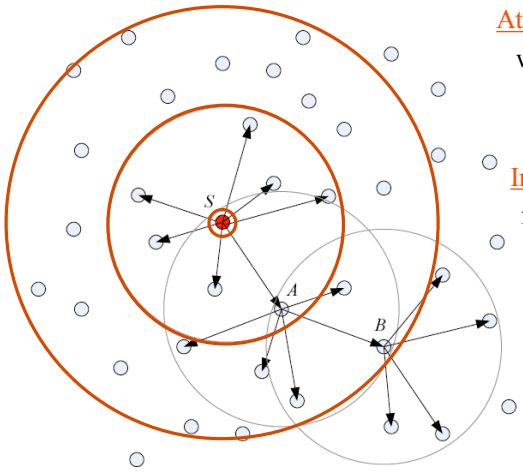
- A broadcast packet is usually forwarded multiple times before reaching all the nodes



NC STATE UNIVERSITY 7

DoS Attacks against Broadcast Authentication (1)

- Signature-based broadcast authentication



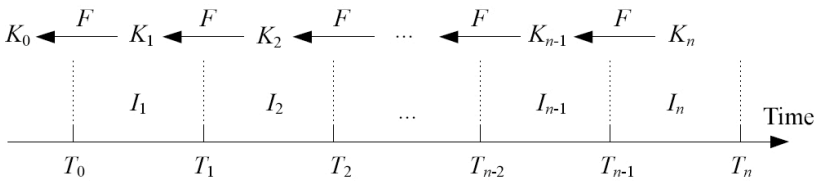
Attack: inject messages with false signatures

Impact: Nodes receiving false messages have to perform unnecessary signature verifications

NC STATE UNIVERSITY 8

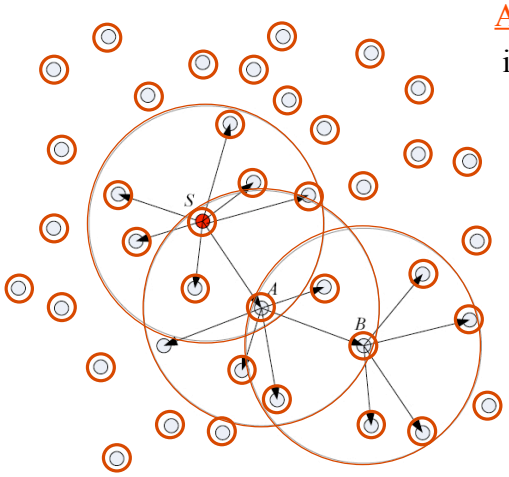
DoS Attacks against Broadcast Authentication (2)

- μTESLA-based broadcast authentication
  - Symmetric cryptography
  - Delayed disclosure of symmetric keys
  - Security condition
    - Make sure when a receiver gets a broadcast message, the sender has not disclosed the corresponding key
    - Loose time synchronization



DoS Attacks against Broadcast Authentication (3)

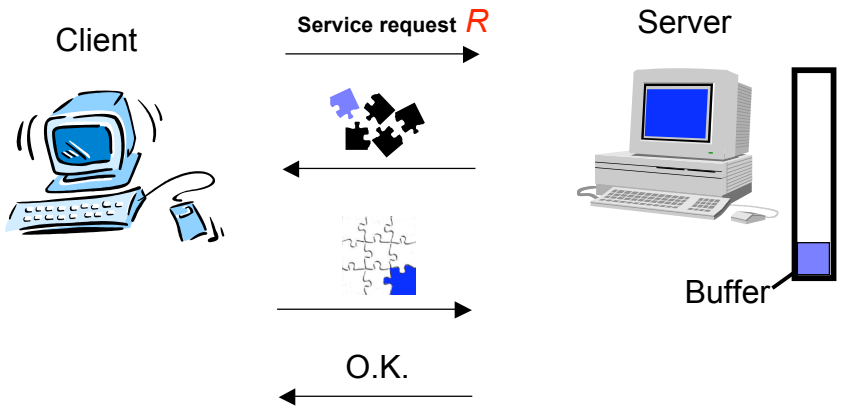
- μTESLA-based broadcast authentication



Attack: Overhear and inject false messages

Impact: All the nodes have to *buffer*, and some have to *forward* the false messages

Possible Solution -- Client Puzzle?



[Juels & Brainard, NDSS '99]

Challenge: Client puzzle requires interaction, but broadcast is one-to-many!

Our Solution: Message Specific Puzzles

- Weak authentication along with signature or μTESLA authentication
  - Extremely efficient to verify, but costly to forge
  - Cannot be pre-computed
  - Even if extremely resourceful attackers may forge a few, but it's very difficult to reuse the forged ones
  - Work for both digital signature and μTESLA
  - Light communication overhead
    - E.g., 4 bytes/packet for μTESLA with reasonable strength
  - Require resourceful sender (e.g., laptop)
  - Sender side delay
    - Still useful when sender can predict the broadcast messages
    - Examples: task dissemination, sensor/actuator reconfiguration

Outline

- Assumptions
- A strawman approach
- Message specific puzzles
- Optimization for signature-based broadcast authentication
- Optimization for μTESLA-based broadcast authentication
- Implementation and field experiments
  - *TinySigGuard* and *TinyμTESLAGuard*

Assumptions

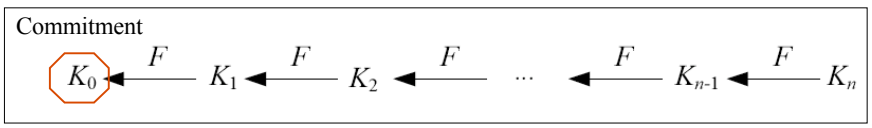
- Broadcast senders are resourceful
  - E.g., laptop, vehicle carried computers
- When signature is used, assume
  - packet is large enough to accommodate the signature
    - Not a problem for ZigBee compliant sensor nodes (up to 102 byte payload)
  - Nodes can perform limited public key operations
- When μTESLA is used, assume the clocks of all nodes are loosely synchronized

Assumptions of Attackers

- Attacker can eavesdrop, inject, and modify broadcast packets
- Attacker is resourceful
- Attacker may use colluding nodes distributed in the network
  - Share information
  - Wormhole
- Attacker may compromise some nodes
  - But attacker cannot compromise the broadcast sender

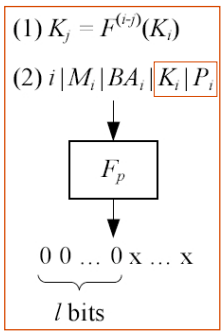
A Strawman Approach

- Weak authentication through one-way key chains
  - The *i*-th broadcast packet includes a weak authenticator K_i.
  - A broadcast packet can be weakly authenticated with the weak authenticator
  - Weaknesses
    - An attacker overhearing a valid broadcast packet can forge many packets
    - An attacker can force the nodes isolated from the source to verify an infinite number of signatures
  - Key problem: Duplicating weak authenticators is too easy

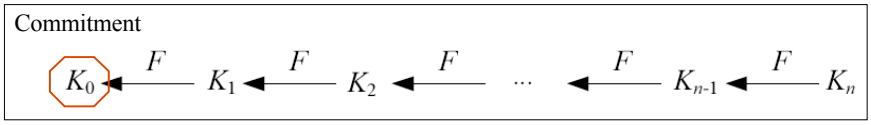


Message Specific Puzzles (MSP)

- Consider the i-th message M_i
  - Suppose BA_i is the signature or μTESLA MAC for the packet w/o weak authenticator
  - K_i is the puzzle key for M_i
  - i, M_i, BA_i, K_i together define a message specific puzzle
  - The solution P_i must satisfy $\Rightarrow$
  - K_i and P_i form the weak authenticator



l: puzzle strength

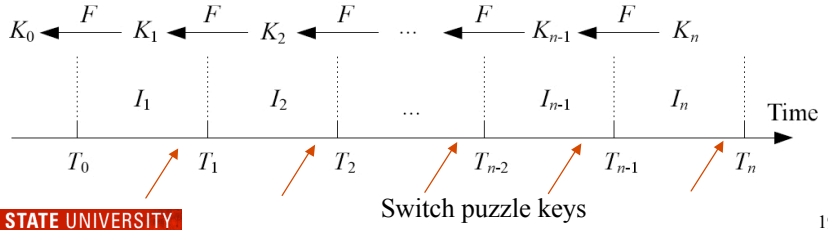


MSP for Weak Authentication with Signature-Based Broadcast Authentication

- Sender
  - Generate a signature BA_i
  - Find the right puzzle key, and solve the MSP for M_i
    - Expensive
  - Broadcast the message index, the message, the broadcast authenticator, the puzzle key, and the puzzle solution
    - $i \mid M_i \mid BA_i \mid K_i \mid P_i$
- Receiver
  - Verify the puzzle solution
    - Very efficient (1 hash operation)
  - Verify the broadcast authenticator
  - Later messages with the same index can be dropped

MSP for Weak Authentication with μTESLA-Based Broadcast Authentication

- Sender
  1. Find the right μTESLA key to generate a MAC (BA_i)
  2. Find the right puzzle key, and estimate how many puzzle solutions can be tested before the μTESLA key expires
     - Try the estimated number of puzzle solutions
  3. If puzzle solution is not found, go back to step 1
  4. Broadcast the message index, the message, the broadcast authenticator, the puzzle key, and the puzzle solution



MSP for Weak Authentication w/ μTESLA

- Receiver
  - Verify the puzzle solution
    - Very efficient (1 hash operation)
  - Verify the broadcast authenticator
    - Have to wait until the μTESLA key is disclosed

Security

- Can an attacker forge a weak authenticator?
  - Yes, if it's resourceful, but
    - The attacker has to wait for the release of the puzzle key, and
    - It takes time
  - Each message has a unique message index
    - Signature: After getting an authenticated message, a receiver will not accept another one with the same index
    - μTESLA: Limited duration for forged messages
    - Problem: network partition

Minimizing Reuse of Forged Puzzle Solutions

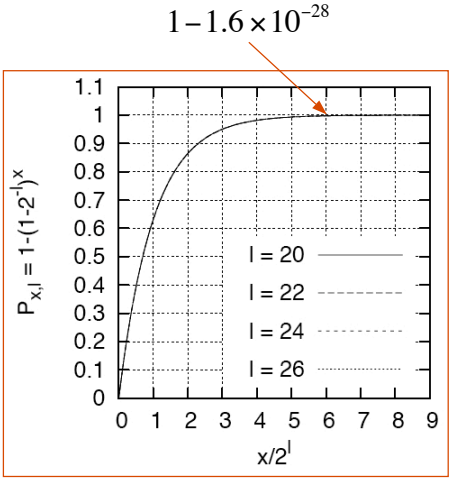
- Forged puzzle solution
  - Valid puzzle solution, but invalid broadcast authenticator
- Buffer hashes of potentially forged puzzle solutions
  - Keep m buffer entries
  - For the k-th (unique) message that pass the weak authentication
    - Discard if it hashes to one of the saved entries
    - If $k <= m$, save it to an empty entry
    - If $k > m$, substitute it for any random entry with probability m/k

Cost of Finding a Puzzle Solution

- Given puzzle strength l, the probability of finding a solution after x trials:

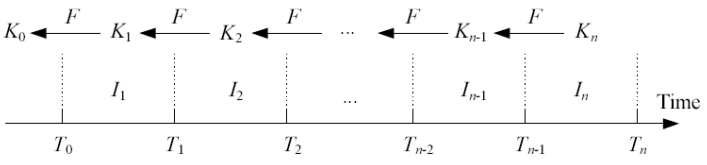
$$P_{x,l} = 1 - (1 - 2^{-l})^x$$

- Expected number of trials to find a solution is 2^l



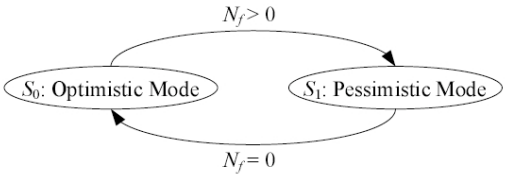
Optimization for Digital Signatures (1)

- Time limited message specific puzzles
  - Adopting the idea of TESLA
  - Addressing the network partition problem
    - An attacker can DoS attack nodes isolated from the source for a limited period of time



Optimization for Digital Signatures (2)

- Adaptive verification
  - N_f: # failed signature verifications in the past w time units
    - An indicator of DoS attacks
  - Optimistic mode
    - Verify the weak authenticator, rebroadcast, and then verify the signature
  - Pessimistic mode
    - Verify the weak authenticator and the signature, and then rebroadcast



Optimization for µTESLA

- Use µTESLA key chain for both weak authentication and broadcast authentication
  - The disclosed key ⇒ puzzle key
  - Reduced complexity
    - Only manage one key chain
  - Reduced computation, communication, and storage overheads

Implementation

- Implemented on TinyOS
  - Target platform: MICAz
  - Assume up to 102 byte packet payload size
    - ZigBee compliant
- Two packages
  - TinySigGuard
    - Intended to be used with TinyECC to provide signature-based broadcast authentication
  - TinyμTESLAGuard
    - Intended to be used with μTESLA-based broadcast authentication
  - Both have a sender part (running on PC) and a receiver part (running on motes)

Implementation (Cont'd)

- Code size measured on MICAz
  - Excluding TinyECC and μTESLA
  - Obtained using check_size.pl

Code size (bytes) on MICAz

	ROM	RAM
TSGReceiver	1,317	289
TuGReceiver	180	210

- Packet format

Signature-based:

$i(2)$	$M_i(up\ to\ 48)$	$Sig(40)$	$K_i(8)$	$P_i(4)$

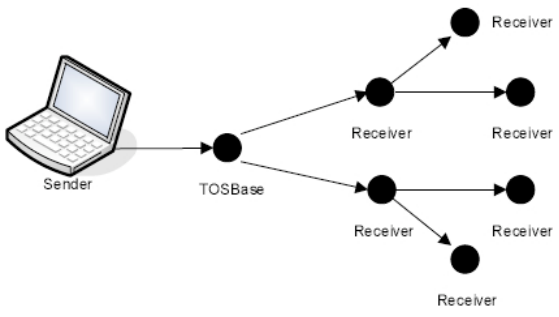
μTESLA-based:

$i(2)$	$j(2)$	$M_i(up\ to\ 78)$	$MAC_{j+d}(8)$	$K_j(8)$	$P_i(4)$

Experimental Evaluation

- Sender
  - DELL Latitude D510 laptop
  - 1.6GHz Pentium M 730
  - 512 MB DDR SDRAM
- Receivers
  - 30 MICAz motes
  - Each has a 8-bit processor, 128KB program flash, 4KB SRAM, ZigBee compliant radio
- Deployed in a 2,800 square-feet lab

Experimental Scenario



Computation Cost and Sender Side Delay

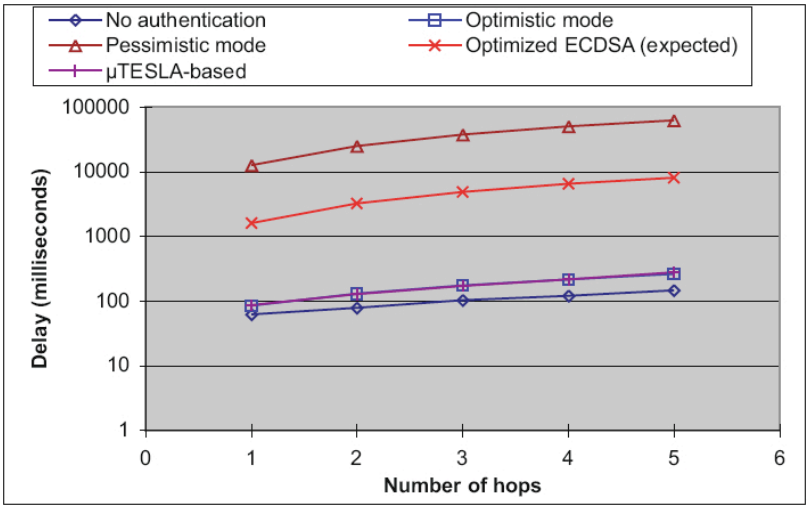
Table II. Expected time (millisecond) required for solving and verifying a message specific puzzle

Puzzle Strength (l)	signature-based		Verifying a Solution (MICAz)
	Java	Crypto++	
20	8,357	2,590	14.6
22	35,220	10,377	14.6
24	142,237	41,440	14.6
26	599,953	165,893	14.6

Table III. Average sender side delay (millisecond) for μTESLA-based broadcast authentication

Puzzle Strength (l)	Duration of each μTESLA interval			
	500 ms	1000 ms	2000 ms	4000 ms
20	7,976	7,337	7,265	6,033
22	26,115	26,073	25,929	25,390
24	105,231	105,219	104,922	104,842
26	431,359	431,031	430,438	429,688

Propagation Delay



Conclusion

- Message specific puzzles
  - Benefits
    - Extremely efficient to verify, but costly to forge
    - Cannot be pre-computed
    - Forged weak authenticators are difficult to reuse Work for both digital signature and μTESLA
    - Light communication overhead
  - Limitations
    - Require resourceful sender (e.g., laptop)
    - Sender side delay

Thank You!