TinySec: Security for TinyOS

Chris Karlof       Naveen Sastry
David Wagner
January 15, 2003

http://www.cs.berkeley.edu/~nks/tinysec
Security Risks in Wireless Sensory Networks

- Eavesdropping
  - Confidentiality
- Packet Injection
  - Access control
  - Integrity
- Jamming
- Replay
- Denial of Service

TinySec
TinySec Architectural Features

- Single shared global cryptographic key
- Link layer encryption and integrity protection → transparent to applications
  - New radio stack based on original
- Cryptography based on a block cipher
Prior Wireless Security Schemes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>802.11b / WEP</td>
<td>Weaknesses found</td>
</tr>
<tr>
<td>GSM</td>
<td>Weaknesses found</td>
</tr>
<tr>
<td>IP / IPSEC</td>
<td>Still secure</td>
</tr>
</tbody>
</table>

- Sensor networks have tighter resource requirements
- Both 802.11 & GSM use stream ciphers
  - Tricky to use correctly
  - Designed by non-cryptographers, in a committee
TinySec Summary

• Security properties
  – Access control
  – Integrity
  – Confidentiality

• Performance
  – 5 bytes / packet overhead
  – Peak bandwidth (8 bytes data):
    25 packets/sec vs. 40 packets/sec
    (TinySec) (MHSR)
Block Ciphers

• Pseudorandom permutation (invertible)
  – DES, RC5, Skipjack, AES
  – Maps $n$ bits of plaintext to $n$ bits of ciphertext

\[ E_k : \{0,1 \}^n \xrightarrow{K = \{0,1 \}^k} \{0,1 \}^n \]

• Used to build encryption schemes and message authentication codes (MAC)
Packet Format

- Key Differences
  - Encrypted
  - No CRC
    - No group ID
    - MAC
    - IV
  - MAC’ed
  - Total:
    - +5 bytes
TinySec Interfaces

- TinySec
  - TinySecM: bridges radio stack and crypto libraries

- BlockCipher
  - 3 Implementations: RC5M, SkipJackM, IdentityCipher
    (SRI has implemented AES)

- BlockCipherMode
  - CBCModeM: handles *cipher text stealing*
    - No length expansion when encrypting data

- MAC
  - CBCMACM: computes message authentication code using CBC
Security Analysis

• Access control and integrity
  – Probability of blind MAC forgery $1/2^{32}$
  – Replay protection not provided, but can be done better at higher layers

• Confidentiality – Reused IVs can leak information
  – IV reuse will occur after $2^{16}$ messages from each node
    [1 msg / min for 45 days]
  – Solutions
    • increase IV length $\rightarrow$ adds packet overhead
    • key update protocol $\rightarrow$ adds complexity
  – Applications have different confidentiality requirements
    • Need a mechanism to easily quantify and configure confidentiality guarantees
  – Applications may provide IVs implicitly
    • Apps may be able to guarantee sufficient variability in their messages
      (eg through timestamps)
## Cipher Performance

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Block Operation Time (cycles)</th>
<th>Block Operation Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC5 C only</td>
<td>~5750 +</td>
<td>1.70 ms</td>
</tr>
<tr>
<td>RC5 SPINS: C + asm</td>
<td>~2775 avg</td>
<td>0.75 ms</td>
</tr>
<tr>
<td>SkipJack TinySec: C only</td>
<td>~2500</td>
<td>0.70 ms</td>
</tr>
<tr>
<td>RC5 TinySec: C + asm</td>
<td>~1775 avg</td>
<td><strong>0.50 ms</strong></td>
</tr>
</tbody>
</table>

- 2 block cipher operations per block, which take 1.0 ms/block
- For comparison, takes an ~4.8 ms to send one block over radio
- Encryption and MAC can be overlapped with transmission/reception
Discussion

• Short packets have more overhead
  – Min data size is 8 bytes (size of block cipher)
  – Packet length not affected for more than 8 bytes of data

• Acknowledgments can be authenticated with no extra work or overhead
  – 1/256 chance of forgery

• Group ID no longer supported
Usage:
How does this change my life?

• Need to be aware of keys & keyfile
  – Currently, keys part of program, not intrinsic to mote (similar to moteID)
  – Plan to use EEPROM to tie key to mote
  – Makerules generates a keyfile if none exists and then uses it for programming all motes;
  – Keyfiles resides in user’s home directory. Manual transfer needed to install motes from different computers.

• Only application level code change:
  – Just use SecureGenericComm instead of GenericComm

• Works in simulator

• Who is using it?
  – Pursuer-Evader demo
  – Bosch
Conclusions

- TinySec can provide transparent security for applications
  - Access control
  - Integrity
  - Confidentiality
- Problems not addressed:
  - Jamming
  - Node or key compromise
  - Replay
  - Denial of service
- Future work
  - Performance improvements
  - Finer granularity key management
  - Replay protection
Extra slides
Symmetric key encryption

- Confidentiality achieved by encryption
- Encryption schemes (modes) can be built using block ciphers
  - CBC-mode: break a $m$ bit message into 64 bit chunks ($m_1,m_2,...$)
  - Transmit ($c_1, c_2, ...$) and $iv$
- $iv$ is needed to achieve semantic security
  - A message looks different every time it is encrypted
  - $iv$ reuse may leak information
Message Authentication Codes

• Encryption is not enough to ensure message integrity
  – Receiver cannot detect changes in the ciphertext
  – Resulting plaintext will still be valid

• Integrity achieved by a message authentication code
  – A $t$ bit cryptographic checksum with a $k$ bit key from an $m$ bit message

$$\text{MAC} : \{0,1\}^m \xrightarrow[K=\{0,1\}^k]{K=\{0,1\}^k} \{0,1\}^t$$
  – Can detect both malicious changes and random errors
  – Replaces CRC
  – Can be built using a block cipher
  – MAC key should be different than encryption key

![CBC-MAC Mode Diagram](image-url)