TinySec : TinyOS Link Layer Security Proposal - version 1.0

Chris Karlof Naveen Sastry Umesh Shankar David Wagner
{ckarlof,nks,ushankar,daw}@cs.berkeley.edu
University of California, Berkeley

July 17, 2002

1 Introduction

1.1 Goals

This is a proposal for a security architecture for link layer communication in TinyOS [1], an event-driven operating system for sensor networks. Our goals are the following:

1. **Provide a baseline security architecture:** Although some applications with strong security requirements (alarm systems, military applications, privacy invading applications, etc) will require additional mechanisms, TinySec's foremost goal is to provide an implementation that will serve the needs of most applications. The perceived priority of those needs is discussed further below.

2. **On by default:** A security mechanism is least useful when it is not used. Users may forget to enable security if it is shipped with it turned off.

3. **Transparent:** Users will quickly ask how to disable security if it is difficult manage or configure. TinySec should require little or no attention from the user during both network configuration and operation.

4. **Granular:** Users should be able to choose tradeoffs between power usage and security strength in a straightforward way. It is our belief that for the majority of sensor networks applications, message integrity and access control are more important than confidentiality, so we offer a very low-overhead version satisfying those needs. However, we believe it is important to make it easy to increase security for more sensitive applications.

5. **Composable:** TinySec operates at the link layer and is intended to compose well with routing protocols and routing security mechanisms. TinySec is purposely lightweight to avoid duplicating functionality more effectively provided by security mechanisms at the network layer. We provide some guidance in this document as to how these layers might interact.

1.2 Assumptions and Regimes

Due to their limited processing and power capacity, particular sensor networks tend to perform a single purpose (run a single application). As such, it is difficult to prescribe a single security model for all applications. We therefore describe different regimes in which sensor networks might be used and the likely security tradeoffs in each.

Here is a list of some relevant parameters to consider:

- **Data rate** Due to power constraints (battery life or environmental recharge rate), no node can send at a high rate for long periods of time. However, depending on the application and routing protocol, traffic may be more or less "bursty". For example, a simple temperature-measuring network would have very predictable, periodic traffic patterns. By contrast, a burglar alarm system might lie largely dormant until some activity was detected, then switch into an active state with constant monitoring for some period. More bursty networks are more problematic, since during heavy traffic periods, local buffer (RAM) space and processing time are likely to be at a premium—likely just when security is needed most.
- **Node Density** There are competing constraints when it comes to node density. On the one hand, in a high-density network, a smaller proportion of nodes are needed to forward packets, thus saving energy. From a security perspective, however, the more neighbors a node receives packets from, in general the more transient state it must maintain. In the highly resource constrained environment of a sensor node (e.g., 4k bytes RAM) this can be a serious problem.
- **Per-message cost vs. Per-byte cost vs. Computation cost** A sensor must report its readings back to the base station, but may in some cases aggregate many readings into one message. Such choices depend on the relative costs of sending a byte on the network, fixed per-message costs, and the cost of performing computation. In-network processing also requires intermediary nodes to be able to process the data. For all but the most security critical applications, this rules out keying mechanisms in which each node exclusively shares a key with the base station.

2 TinySec

2.1 Security Properties

The details of the protocol are given in section 2.2; in this section we summarize relevant security properties,

Access Control Unauthorized nodes should not be able to participate in the network by either acting as a router or injecting new traffic. See notes regarding replay in section 2.4.

Message integrity Adversaries should not be able to alter existing messages. This property is satisfied by the use of a MAC over each packet.

Confidentiality The data portion of each packet is encrypted; however, the length of the packet and the recipient's address are not obscured. See section 2.3 for details regarding the level of protection provided.

Effect of node compromise TinySec uses a globally shared key. The compromise of one node will compromise the entire network.

2.2 TinySec specification

2.2.1 TinyOS packet format

TinyOS packets are currently at most 36 bytes long and have the following format:

TinyOS packet format

Field	Length
Destination ID	2 bytes
Active message handler	1 byte
Group ID	1 byte
Data length	1 byte
Data	29 bytes (max)
CRC	2 bytes

Notes: Destination ID is the node ID of the next hop. The final destination is implied to be the base station. The Group ID is used prevent interference between different sensor networks or create groups within a single sensor network. Data payload can be from 0 to 29 bytes, and its length is indicated in the data length field.

If the CRC passes, a 1 byte acknowledgement is immediately sent to the sender. This acknowledgement currently contains no useful information.

2.2.2 TinySec packet format

In the TinyOS packet format, the group ID (1 byte) and CRC (2 bytes) collectively provide some level of access control and error detection. We propose replacing these with a 3 byte MAC (CBC-MAC). For encryption we use RC5, since it is efficient on sensor hardware [2]; a single globally shared key is used among the network group (the entities that used to share group IDs).

Basic TinySec packet format

Field	Length
MAC	3 bytes
Destination ID	2 bytes
Data length	1 byte
Active message handler	1 byte
Encrypted data	29 bytes (max)

For message transmission, if ID is the destination ID, AM is the Active message handler, L is the data length, and D is the data payload, the packet sent over the radio is $(MAC, ID, L, AM, \text{Enc}_{K_e}(AM, D))$ where $MAC = \text{CBC-MAC}_{K_m}(ID, L, AM, D)$ and K_e and K_m represent globally shared keys for encryption and message authentication. In Enc we propose using RC5-32/8/16 (64-bit blocks, 8 rounds, 128-bit key) in tweaked-CBC mode (RC5 is first applied to the IV before being XOR'ed with the first plaintext block) with (ID, MAC) as an IV for each message. Ciphertext stealing is used to ensure that the length of the transmitted ciphertext is the exact length of the transmitted plaintext (as opposed to padding the last plaintext block to the block length boundary).

CBC-MAC is only secure for fixed length messages, although there exist techniques for making it secure for variable length messages [3].

2.2.3 Notes on the packet format

The MAC is on the underlying plaintext, which requires decryption of the packet before the MAC can be checked. Since acknowledgement packets must be sent fairly soon after reception, there may be some timing issues regarding how fast this authentication can be performed. Another option is to make the MAC over the encrypted data. Now decryption can be performed after the acknowledgement has been sent, but we lose the ability to use the MAC as part of the IV. Note that the MAC must the near the beginning of the packet because it is needed for decryption.

In order to prevent acknowledgement spoofing, we propose making the acknowledgement byte sent a MAC of the MAC in the transmitted packet (details to come) [More thoughts on this?].

2.3 Semantic security

Semantic security ensures an eavesdropping adversary can obtain no information about the plaintext, even if it sees multiple encryptions of the same plaintext. One common method of achieving this in symmetric cryptography is to use an Initial Value (IV) in the encryption function; this value may be sent with the message or kept implicitly by both parties in the form of a counter or the clock value.

TinySec does not provide semantic security. Multiple encryptions of the same plaintext result in the same ciphertext. This makes traffic analysis of binary messages sent to the same node easy. However, the inclusion of the MAC in the IV guarantees that a single bit change in underlying plaintext will effect the entire resulting ciphertext.

TinySec sacrifices semantic security to save power. Not transmitting an explicit IV in each packet saves several bytes per packet, resulting in significant power savings over the lifetime of the network. We realize that some applications may require stronger confidentiality and provide the option to include an explicit IV as part of the data payload.

2.4 Replay Protection

This means that an adversary can not replay earlier messages (attempting to pass them off as current transmissions) without being detected. If a counter or clock is used as the IV, it is easy to check that the IV is not being reused. The downside is that one must keep state for the counter value (for each link or destination) or use a time synchronization protocol.

TinySec does not provide replay protection. Section 3.1 discusses methods (and the corresponding tradeoffs) for defending against replay. It is not clear that replay protection is the responsibility of a link layer security module. Replay protection requires a receiver to maintain state for each neighbor (or sender), and it would be wasteful to duplicate any state maintained by a routing algorithm regarding a node's neighbors. In the interest of keeping TinySec stateless, simple, and composable, we believe replay protection should most likely the responsibility of the network layer, although it is conceivable these two layers might be highly integrated for some particularly resource constrained applications.

2.5 Key management

During deployment, all nodes are loaded with a master key K. Separate encryption and MAC keys are derived using a pseudo-random function applied to the master key, i.e., $(K_e, K_m) = PRF(K, s)$ where s is some salt. s can be periodically updated and broadcast to all the nodes by the base station, or with very loose time synchronization, s can be a counter that is periodically incremented.

2.6 Random Number Generation

Obtaining a true random number source can often prove to be a challenge. We note that the nodes are all endowed with sensor capabilities as this is one of their primary functions. Thus, we propose to use the low order bits of the sensors to provide entropy to feed a cryptographically strong random number generator. We expect this facility to be useful for the protocols outlined here as well as others in the system – for example in packet collision backoff timeouts.

3 Making TinySec stronger

3.1 Security vs. Power and RAM Tradeoffs

The baseline TinySec protocol provides a baseline confidentiality and integrity guarantee with a low computation overhead and almost no communication overhead, but the level of security provided may not be sufficient for all applications. In the table below we describe some additional security guarantees and mechanisms by which they can be achieved. These features come at a price (naturally): they all consume more of at least one of the two most precious resources on a sensor node: RAM and battery energy. Some require cooperation from other layers of the networking stack.

Security tradeoffs

If you want:	**Then add:**	**Issues:**
Jamming protection	Hardware solution (spread spectrum)	Might consume battery power or add computational complexity; limits bit-rate; not upgradeable
Semantic security	Random IV in packet	Per-packet overhead; does not prevent replays; can be included along with app-level data
Semantic security	Implicit counter used as IV for each packet	Pairs of nodes maintain a pair of counters that is incremented for each received packet; in the presence of packet loss, successive counter values are tried until successful decryption; requires keeping state for each neighbor; questionable performance in the presence of high loss rates.
Replay protection + semantic security	Explicit counter as IV	Requires keeping state for each neighbor; per-packet overhead; RED-like techniques to get probabilistic guarantee against replay using constant space
Replay protection + semantic security	Use clock as IV	Per-packet overhead; requires local or global time synchronization (more messages to setup); given a synchronization error of Δ, refuse messages older than current time $-$ prop. time $-$ Δ; if Δ is high, effectiveness is limited; no additional local state needed.

3.2 Keying mechanisms

Below we present a possible keying mechanisms and discuss the tradeoffs of using them in sensor networks. The tradeoffs focus on two important mechanisms frequently used in sensor networks, *in-network processing* and *passive participation*. Nodes will often aggregate data from several received packets and forward a single packet containing the aggregated values. This helps reduce redundancies in sensor readings, message traffic, and power consumption. Passive participation is a special form of in-network processing. Nodes normally only aggregate data readings in packets specifically addressed to it, but by using passive participation, nodes may taken some action based on *overheard* message traffic between its neighbors. Particular keying mechanisms may reduce the effectiveness or eliminate the possibility of in-network processing and passive participation.

Keying mechanisms

Keying mechanism:	**Benefits:**	**Tradeoffs:**
Shared global key	Simple; does not prohibit in-network processing or passive participation.	The compromise of a single node will compromise the entire network.
Per-node keys shared with base station and shared global key	Simple; does not prohibit in-network processing or passive participation; neighbors can be authenticated via base station.	Node compromise is still an issue, but neighbor authentication provides additional protection.
Per-node keys shared with base station	Compromised nodes cannot alter nor eavesdrop on other nodes' messages to the base station; neighbors can be authenticated via the base station	Prohibits in-network processing and passive participation; message integrity cannot be checked until receipt at base station, thus increasing effectiveness of external resource consumption attacks.
Per-node keys shared with base station and shared global MAC key	Message integrity and access control can be enforced at intermediate nodes; Compromised nodes cannot alter nor eavesdrop on other nodes' messages; neighbors can be authenticated via base station.	Prohibits in-network processing and passive participation.
Per-node keys shared with base station and per-neighbor keys set up via base station [2]	In-network processing possible; a compromised node cannot modify or eavesdrop on messages not addressed to it; neighbors can be authenticated via the base station; neighbor authentication and key set-up can be piggybacked.	Passive participation not possible; message overhead for key set-up.
Per-neighbor keys established using key infection and shared global key	In-network processing and (some level of) passive participation possible; reduced reliance on base station; shared global key still allows access control and confidentiality against external adversaries; local communication for per-neighbor key set-up.	Sufficiently powerful compromised nodes may be able to eavesdrop on and modify messages all over the network, but if key infection done quickly at network deployment, the likelihood of this threat is reduced; neighbor authentication still requires base station involvement.

Notes:

- Not all routing algorithms have good mechanisms for the base station to address individual nodes. A powerful base station able to reach most nodes in a single can help with this.
- Since bandwidth in the links surrounding the base station is especially precious, any key set-up and

authentication protocols involving the base station are prone to DOS attacks and may be used themselves in mounting DOS attacks.

- Key infection can be piggybacked on local time synchronization.

References

[1] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, "System architecture directions for networked sensors," in *Proceedings of ACM ASPLOS IX*, November 2000.

[2] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J.D. Tygar, "Spins: Security protocols for sensor networks," in *Proceedings of Mobile Networking and Computing 2001*, 2001.

[3] Mihir Bellare, Joe Kilian, and Phillip Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61, no. 3, pp. 362–399, 2000.