Handling Reboots and Mobility in 802.15.4 Security

Eirik Klevstad

February 9, 2016

1 Abstract

To survive reboots: Store anti-replay data, frame counters in non-volatile memory.

For IoT nodes: Flash memory = Energy consuming and slow.

Session keys frees nodes from storing anti-replay data and frame counters in flash memory.

APKES suggested. Does not support reboots nor mobile nodes.

AKES: Survives reboots without storing data in non-volatile memory.

Independent: Frame counter reaches maximum value.

2 Introduction

802.15.4 networks where nodes communicate with each other over IPv6: 6LoWPAN.

Suitable for smart cities, industrial monitoring, precision agriculture (smart jordbruk)

802.15.4 security filters out injected and replayed packets by using MIC and an incrementing frame counter.

Optional: encryption

Uses tweaked Counter with CBC-MIC (CCM) that uses AES-128.

Problematic to keep track of all neighbours' frame counter due to limited RAM space.

For large, mobile networks: Some of the information needed to enforce anti-replay needs to be swapped to non-volatile memory (flash).

Flash memory: Energy consuming and slow.

Routing traffic between non-neighbouring nodes (tunnelling) = hidden wormholes. Nodes are lured into believing that they are neighbours and needs to store information about fictional neighbours forever.

To survive reboots: All anti-replay data must be stored in non-volatile memory.

Problems if not: Nonce reuse in 802.15.4 security standard, and neighbours considers the rebooted nodes' frames as replayed.

Session keys to the rescue: Obivate need for swapping, as well as fix all the three issues with rebooting.

Session keys invalidates MICs from previous sessions, which allows nodes to delete anti-replay data about nodes that have disappeared. Replayed frames from previous sessions are filtered out. Frame counter can also start over without any frames being discarded.

Because: reusing nonces with new session keys is secure.

Four limitations of APKES:

1) Delete anti-replay data about disappeared nodes is allowed, but never done.

2) APKES ignores HELLOs from current neighbours. Causes deadlock after reboot.

3) Broadcasts HELLO only at startup - No functionality for node discovering.

4) Focus on deriving session keys from pre-distributed pairwise keys. Lack of support for pre-distributed network-wide keys.

AKES fixes:

1) Pings neighbours to see if they are still there, if not: Delete.

2) Processes HELLOs from current neighbours as well.

3) Uses Trickle to discover neighbours at runtime.

4) Pluggable schemes still available, but AKES adds the network-wide key scheme.

3 Related Work

3.1 Establishing Pairwise Session Keys

802.15.4: Key establishment unspecified.

Pairwise keys better in case a node is compromised, and to detect compromised nodes.

Pairwise session keys using Public Key Crypto: Heavy and slow in 802.15.4 nodes. Key Distribution Centers are even more heavy.

Pairwise key predistribution schemes = good. Nodes preloaded with material to establish pairwise keys.

Fully pairwise keys: Preload each node with n-1 pairwise keys. high memory consumption.

3.2 Avoiding Swapping

Bloom filters considered for storing anti-replay data, but contains false negatives.

Another solution: 802.15.4e: Timeslotted Channel Hopping (TSCH) media access (MAC) protocol. Does not work with rebooting and compromises.

3.3 Authenticating Broadcast Frames

Pairwise session keys can authenticate for unicast frames, but not broadcast. Can use group session keys, but sacrifices compromise resilience. EBEAP proposed.

EBEAP concatenates the message MIC'ed multiple times with each neighbours pairwise session key.

Then sends the real deal, which is used to generate the MIC and compare with the previous MIC (buffered in a ring buffer). If fresh and match: accept frame.

4 AKES: Adaptive Key Establishment Scheme

Uses pluggable key pre-distribution scheme.

Bases pairwise session keys on preshared secrets.

4.1 Preloaded Configuration Settings

AKES nodes must be preloaded with addressing information and keying material.

4.1.1 Addressing Information

Each node: 2-byte personal area network (PAN) and variable-size address.

PAN ID identifies all nodes that belong a certain PAN. Can be used to separate co-located 802.15.4 networks or divide them into sub-networks.

Three different addresses: 8-byte extended (globally unique), 2-byte short(PAN unique), 1-byte simple (PAN unique)

AKES reuses 802.15.4 addressing scheme. When requesting shared secret of node v, the request contains both PAN ID and the address of v to the plugged-in key pre-distribution scheme.

Two restrictions on use of addresses:

1) Can not change address during a session.

2) Does not work with protocols for auto-configuring PAN identifiers, short, and simple addresses, which requires 802.15.4 security (AKES has not reached that point at this time)

Therefore: Addressing needed at runtime must be preloaded in the node.

4.1.2 Keying Material

Node must be preloaded with any keying material, which is specific to the plugged-in key pre-distributed scheme.

Supports: Fully pairwise, network-wide, Blom's Scheme, random pairwise.

4.2 Establishing Session Keys

Three-way handshake to establish session keys.

Command frames: HELLO, HELLOACK, ACK. Processed at link layer.

Same stuff as in the main paper.

Crucial change to APKES: AKES handles HELLO from permanent neighbours, which was discarded in APKES, to support rebooting.

Store node both as temporal and permanent neighbour until receiving ACK, then delete permanent and convert temporal = New session.

4.3 Handling Mobility

Trickling to be able to deal with a changing neighbourhood.

4.3.1 Deletion of Disappeared Permanent Neighbour

Permanent neighbour expires: AKES checks if the neighbour is still in range. Uses two commands:

(authenticated) UPDATE and UPDATEACK

Prolongs expiration time at both sides.

Deletes if not successful UPDATEACK

4.3.2 Trickle-Based Broadcasting of HELLOs

Challenge: Broadcast as few HELLOs as possible, while quickly reacting to neighbourhood changes.

Trickle: Algorithm for disseminating information in wireless networks.

For each consistent transmission, the counter is incremented until a time t.

5 Security Analysis

6 Implementation

7 Evaluation

8 Conclusions and Future Work