POTR: Practical On-the-fly Rejection of Injected and Replayed 802.15.4 Frames

Konrad-Felix Krentz, Christoph Meinel
Internet Technologies and Systems
Hasso Plattner Institute
Potsdam, Germany
{*konrad-felix.krentz, christoph.meinel*}*@hpi.de*

Maxim Schnjakin
Innovations
Bundesdruckerei
Berlin, Germany
maxim.schnjakin@bdr.de

***Abstract*—The practice of rejecting injected and replayed 802.15.4 frames only after they were received leaves 802.15.4 nodes vulnerable to broadcast and droplet attacks. Basically, in broadcast and droplet attacks, an attacker injects or replays plenty of 802.15.4 frames. As a result, victim 802.15.4 nodes stay in receive mode for extended periods of time and expend their limited energy. He et al. considered embedding one-time passwords in the synchronization headers of 802.15.4 frames so as to avoid that 802.15.4 nodes detect injected and replayed 802.15.4 frames in the first place. However, He et al.'s, as well as similar proposals lack support for broadcast frames and depend on special hardware. In this paper, we propose Practical On-the-fly Rejection (POTR) to reject injected and replayed 802.15.4 frames early during receipt. Unlike previous proposals, POTR supports broadcast frames and can be implemented with many off-the-shelf 802.15.4 transceivers. In fact, we implemented POTR with CC2538 transceivers, as well as integrated POTR into the Contiki operating system. Furthermore, we demonstrate that, compared to using no defense, POTR reduces the time that 802.15.4 nodes stay in receive mode upon receiving an injected or replayed 802.15.4 frame by a factor of up to 16. Beyond that, POTR has a small processing and memory overhead, and incurs no communication overhead.**

***Keywords*-Broadcast attack, droplet attack, denial-of-sleep attack, MAC security, ContikiMAC, 6LoWPAN.**

I. Introduction

Among the most popular media access control (MAC) protocols for 802.15.4 networks is ContikiMAC [1], [2]. In fact, ContikiMAC is the default MAC protocol of Contiki - a widely-used operating system for Internet of things (IoT) devices [3]. Conceptually, ContikiMAC inherits various ideas from other MAC protocols. From B-MAC [4], ContikiMAC inherits the idea of scanning the channel for activity periodically. From WiseMAC [5], ContikiMAC inherits the idea of learning the wake-up times of neighboring nodes. From BoX-MACs [6], ContikiMAC inherits the idea of using frames as wake-up strobes. There are also numerous follow-up efforts, which improve on various aspects of ContikiMAC [7]–[11].

The focus of this paper is on securing ContikiMAC against specific denial-of-sleep attacks, namely broadcast and droplet attacks [12]–[14]. In general, denial-of-sleep attacks cause a high energy consumption by depriving victim nodes of entering a low-power sleep mode [12]. Especially, MAC protocols should be secured against denial-of-sleep attacks since receiving and transmitting are both energy-consuming operations [12]. The way how denial-of-sleep attacks on MAC protocols are carried out depends on the MAC protocol under attack [13]. In ContikiMAC, if a node does not detect any frame it can go back to sleep rather soon, whereas, if a frame comes in, this frame is to be received and processed. Thus, to cause higher energy consumption, an attacker can, e.g., inject or replay a broadcast frame. Such denial-of-sleep attacks are known as broadcast attacks [12], [13]. Alternatively, an attacker can also just inject a synchronization header (SHR) followed by a length indication and then stop transmitting [14]. Receivers will interpret the noise that follows as a frame, just to conclude that the appended Frame Check Sequence (FCS) is invalid. These so-called droplet attacks are especially interesting for attackers that are energy constrained as droplet attacks require transmitting less data than broadcast attacks.

To avoid that 802.15.4 nodes detect injected and replayed 802.15.4 frames in the first place, He et al. considered adopting a technique called frame masking [14], [15]. As per 802.15.4, SHRs of 802.15.4 frames are actually fixed to 0x00000000a7. Nevertheless, CC2420 transceivers support changing an SHR's final two bytes to a custom value [16]. Accordingly, the idea of frame masking is to replace an SHR's final two bytes with a pseudo-random value that is derived from a pairwise key between sender and receiver, as well as the index of the current timeslot. In effect, 802.15.4 nodes solely detect 802.15.4 frames with valid "one-time SHRs".

However, owing to deriving one-time SHRs from pairwise keys, frame masking lacks support for broadcast frames. Moreover, the straightforward solution of replacing pairwise keys with group or network-wide keys does not work out because attackers can then bypass frame masking via a hidden wormhole, as we will detail in Section III-C.

After all, newer CC2538 transceivers do not support changing SHRs [17]. Nonetheless, CC2538, CC2420, as well as virtually all other 802.15.4 transceivers from Texas Instruments support parsing incoming 802.15.4 frames already during receipt. This capability of parsing incoming frames during receipt could be used to reject injected and replayed 802.15.4 frames early during receipt.

Indeed, in this paper, we propose a method that uses this capability to reject injected and replayed 802.15.4 frames early during receipt, named Practical On-the-fly Rejection (POTR). The basic approach of POTR is to embed one-

time passwords (OTPs) in the headers of 802.15.4 frames and to validate them during receipt. Compared to frame masking, POTR has four advantages:

- First, POTR supports broadcast frames while avoiding vulnerabilities posed by hidden wormholes.
- Second, POTR is more interoperable since parsing incoming 802.15.4 frames during receipt is a more common capability of 802.15.4 transceivers than changing the SHRs of 802.15.4 frames.
- Third, in POTR, OTPs need not be configured in advance, thereby freeing receivers from configuring OTPs regardless of whether a frame will come in or not.
- Forth, POTR obviates the need for time synchronization by using frame counters instead. This makes POTR particularly lightweight in terms of processing, memory, and communication overhead.

II. Related Work

Originally, frame masking was intended to prevent reactive jamming attacks [15]. Later, He et al. found that frame masking also prevents droplet attacks [14]. Similar concepts appeared in the context of wake-up receivers [18], [19]. Falk et al. designed a wake-up receiver, which only wakes up upon receiving an OTP that is contained in a list of acceptable OTPs [18]. Yet, in Falk et al.'s design, nodes have to run an energy-consuming initialization protocol and senders and receivers can get out of sync [19]. Aljareh et al. solved both problems via time-synchronized OTPs [19]. POTR expands on these efforts in four directions. First, as opposed to all previous proposals [14], [15], [18], [19], POTR supports broadcast frames. Second, rather than depending on special hardware [14], [15], [18], [19], POTR leverages a common capability of off-the-shelf 802.15.4 transceivers. Third, unlike all previous proposals [14], [15], [18], [19], POTR obviates the need for configuring OTPs in advance. Finally, POTR uses frame counters in lieu of time synchronization. Using frame counters was also conceived by Falk et al. [18], but they take the implicit assumption that senders and receivers are always in sync and neglect, e.g., frame loss. By contrast, POTR tolerates frame loss to a great extent and resynchronizes if a sender and a receiver get out of sync anyway.

More recently, Hsueh et al. secured two other MAC protocols against denial-of-sleep attacks [20], namely X-MAC and A-MAC [21], [22]. In X-MAC, receivers wake up periodically and scan the channel for activity like in ContikiMAC. However, while senders in ContikiMAC repeatedly send a frame until they receive an acknowledgement frame, senders in X-MAC transmit short strobes to wake up receivers. If a receiver detects a strobe, it replies with an acknowledgement frame, whereupon the sender transmits the actual frame. In A-MAC, receivers regularly transmit beacons. Senders wait until they receive a beacon from the intended receiver and thereupon transmit an acknowledgement frame followed by the actual frame. To secure X-MAC and A-MAC against denial-of-sleep attacks, Hsueh et al. suggest exchanging random numbers during the strobe-acknowledgement and beacon-acknowledgement negotiation, respectively. These random numbers are used to derive an OTP, which is then prefixed to the actual frame. During receipt of the actual frame, receivers check if the prefixed OTP is valid and otherwise disable the receive mode immediately. Unfortunately, Hsueh et al.'s method is inapplicable to ContikiMAC since, in ContikiMAC, frames are sent without prior negotiation. Besides, Hsueh et al. only explain the securing of unicast frames and only provide simulation results.

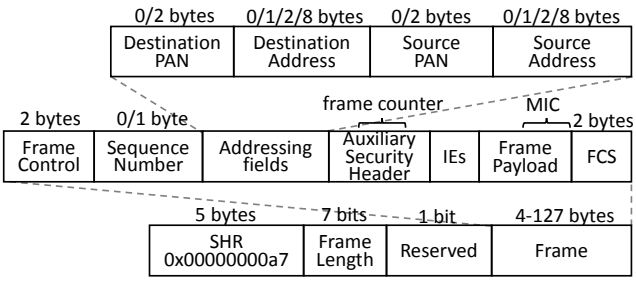

Figure 1: 802.15.4 frame format

Another paradigm of defending against denial-of-sleep attacks is to adaptively react to such attacks. For example, Ren et al. use various statistics to detect different kinds of denial-of-sleep attacks [23]. They suggest switching to a low-power sleep mode in the face of denial-of-sleep attacks for a while. Similarly, Raymond et al. proposed decelerating the wake-up frequency when multiple consecutive inauthentic or replayed broadcast frames come in [24]. A problem with these reactive defenses is that senders are sometimes unaware that a receiver is currently counteracting denial-of-sleep attacks, thus causing an increased energy consumption due to retransmissions. Moreover, intermittently launched broadcast and droplet attacks are hard to detect, whereas proactive defenses, such as POTR, mitigate broadcast and droplet attacks reliably.

III. Background

Since POTR adapts the 802.15.4 frame format and integrates with 802.15.4 security, this section provides a brief introduction to these subjects. Also, we review the motivation behind and vulnerabilities of frame masking.

A. 802.15.4 Frame Format

Figure 1 shows the format of 802.15.4 frames. To indicate the start of an 802.15.4 frame, the physical layer prefixes both an SHR and a Frame Length field to 802.15.4 frames. In addition, the MAC layer adds various other fields to the upper-layer payload. Most importantly for our purposes, each 802.15.4 frame potentially has a destination and a source address. There are three addressing modes available, namely 1-byte simple, 2-byte short, and 8-byte extended addresses. Destination and source addresses are accompanied by a destination and a source personal area network (PAN) identifier, respectively. While simple and short addresses are just PAN unique, extended addresses

are globally unique. The PAN identifier 0xffff is reserved for broadcasting as is the short address 0xffff.

B. 802.15.4 Security

To filter out injected and replayed 802.15.4 frames, 802.15.4 security adds a message integrity code (MIC) and a frame counter to the Frame Payload and the Auxiliary Security Header field, respectively. Receivers shall only accept a frame with an authentic MIC and a frame counter that is greater than that of the last accepted frame from the sender.

However, 802.15.4 security leaves two important mechanisms unspecified, namely session key establishment and the deletion of anti-replay data about neighbors that got out of range. Krentz et al. proposed the Adaptive Key Establishment Scheme (AKES) to fill these gaps [25]. AKES optionally establishes pairwise session keys, group session keys, or both. The session key establishment protocol of AKES is a three-way handshake, as shown in Figure 2. AKES self-adaptively schedules the broadcasting of HELLOs. Receivers that wish to establish session key(s) with a HELLO sender store the sender as a tentative neighbor and, after a random backoff period, reply with a HELLOACK. Upon receipt of a HELLOACK, the HELLO sender checks the validity and authenticity of the HELLOACK and, if successful, stores the HELLOACK sender as a permanent neighbor along with its session key(s) and anti-replay data. Next, the HELLO sender acknowledges with an ACK. As a HELLOACK sender receives an ACK, the HELLOACK sender ensures (i) that the ACK sender is currently stored as a tentative neighbor and (ii) that the ACK is valid and authentic. If both are fulfilled, the HELLOACK sender stores the ACK sender as a permanent neighbor along with its session key(s) and anti-replay data, too. If a permanent neighbor is incommunicado for a certain time, AKES sends an UPDATE to that neighbor. Unless that neighbor replies with an UPDATEACK after a few retransmissions, it will be deleted along with its session key(s) and anti-replay data.

Furthermore, to shorten 802.15.4 frames, Krentz et al. tailored the last bits (LB) optimization to 802.15.4 security [25]–[27]. When using the LB optimization, frames do not carry the entire frame counter. Instead, only the least significant bits are being transmitted and receivers estimate the higher order bits. To make this work, each node has to use a separate per neighbor frame counters for unicast frames and a separate frame counter for broadcast frames. If a node loses track of a permanent neighbor's unicast or broadcast frame counter, AKES triggers an UPDATE-UPDATEACK negotiation for resynchronization.

C. Frame Masking

802.15.4 security does, however, not protect against broadcast and droplet attacks [12]–[14]. The vulnerability that such attacks exploit is that 802.15.4 nodes usually stay in receive mode as long as is indicated by the Frame Length field once they detected an SHR. A mitigation technique is to enable the in-built frame filtering provided

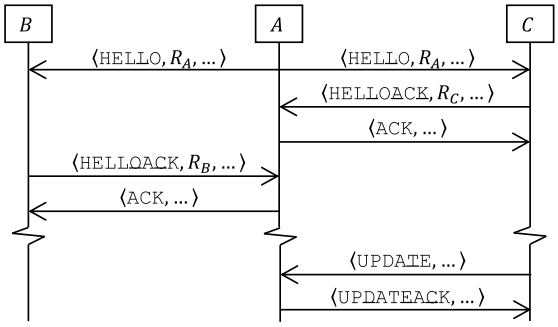

Figure 2: AKES's three-way handshake for session key establishment, and AKES's UPDATE-UPDATEACK negotiation for checking if a node is still in range, as well as for resynchronizing frame counters

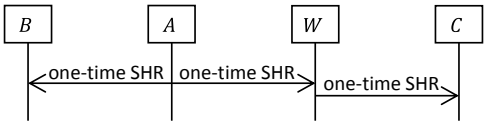

Figure 3: The hidden wormhole W wakes up C

by many 802.15.4 transceivers [14]. For example, the in-built frame filtering of CC2420 and CC2538 transceivers filters out frames that are destined to other nodes by inspecting the Destination PAN and Destination Address field [16], [17]. Yet, attackers can easily bypass such checks via spoofing [14].

As mentioned in the introduction, frame masking avoids the detection of injected and replayed 802.15.4 frames by replacing the final two bytes of SHRs with a pseudo-random value that is derived from a pairwise key between sender and receiver, as well as the index of the current timeslot [14], [15]. That is, instead of scanning the channel for a static SHR, receivers scan the channel for a specific one-time SHR. However, CC2420 transceivers can only scan the channel for a single SHR at a time. This restriction raises three vulnerabilities when trying to add support for broadcast frames to frame masking:

1) Indefinite broadcast frames: First, assume that the receivers of a broadcast frame are indefinite, e.g. during neighbor discovery. Then, the one-time SHR of the broadcast frame should also be derived from a network-wide key. Furthermore, all nodes that currently listen for indefinite broadcast frames should be ready to accept that SHR. Attackers can exploit this by replaying the one-time SHR of the broadcast frame in the same timeslot elsewhere. For example, in Figure 3, the one-time SHR sent from A to B is replayed by W to wake up C. Such attacks are often referred to as hidden wormholes [28].

2) Targeted broadcast frames: Second, if a broadcast frame is only destined to known neighbors $B_1, \ldots, B_n$, one can derive its one-time SHR from a group key so as to limit the scope of the one-time SHR. However, $B_1, \ldots, B_n$ may also wish to accept broadcast frames from their neighbors in the same timeslot and so on. In this case, one-time SHRs of targeted broadcast frames are to be derived from a network-wide key, too. This can, again,

be exploited via a hidden wormhole.

3) Unicast frames: Third, if a node accepts both unicast and broadcast frames in a single timeslot, unicast frames have to have the same one-time SHRs as broadcast frames. Consequently, in Figure 3, an eavesdropped one-time SHR of a unicast frame from A to B can also be misused to wake up C if both C and B accept broadcast frames in this timeslot.

IV. POTR: Practical On-the-fly Rejection

While CC2420 transceivers can only scan the channel for a single SHR at a time, theirs and other 802.15.4 transceivers' capability of parsing incoming 802.15.4 frames during receipt enables performing arbitrary checks. POTR uses this flexibility to avoid all of the above vulnerabilities. Below, we first specify how POTR adapts the 802.15.4 frame format. Subsequently, we explain how POTR integrates with 802.15.4 security, and what POTR does to reject unwanted 802.15.4 frames early during receipt.

A. Adaptation of the 802.15.4 Frame Format

POTR makes six changes to the 802.15.4 frame format, as shown in Figure 4:

- First, POTR replaces the Frame Control field with the Frame Type field. Possible frame types are listed in Table I. The frame type implies which fields follow.
- Second, POTR assumes a network-wide agreement on whether simple, short, or extended addresses are used.
- Third, POTR moves the Source Address field to the beginning of frames.
- Forth, POTR reduces the Auxiliary Security Header field to the Frame Counter field. This works because the Auxiliary Security Header field is unnecessary except for the Frame Counter field therein [25]. Furthermore, when using the LB optimization, it suffices to send the 8 least significant bits of the respective unicast or broadcast frame counter [25].
- Fifth, POTR adds the l-bit OTP field for embedding OTPs in 802.15.4 frames. Actually, the OTP field replaces the Destination Address field since POTR's OTPs encode the intended receiver(s).
- Sixth, POTR renders PAN identifiers redundant, too. This is because the main practical use of PAN identifiers is to reject overheard frames from co-located 802.15.4 networks. However, POTR filters out any unwanted frames early during receipt, including frames from co-located 802.15.4 networks.

Altogether, despite adding the OTP field, POTR incurs no communication overhead by shortening the Frame Control field, as well as removing the Destination PAN, the Source PAN, and the Destination Address field in exchange.

B. Integration with 802.15.4 Security

Besides avoiding communication overhead, POTR also keeps its random-access memory (RAM) overhead low. This is achieved through the integration of POTR with 802.15.4 security. Specifically, POTR reuses the group session keys that AKES establishes for use in 802.15.4 security. Yet, POTR does not use these group session keys "as is", but XORs them with a predistributed network-wide key K_n so as to prevent related-key attacks. Furthermore, POTR takes over the task of checking the freshness of incoming frames from 802.15.4 security and manages all anti-replay data. Hence, 802.15.4 security must disable its own anti-replay protection when using POTR.

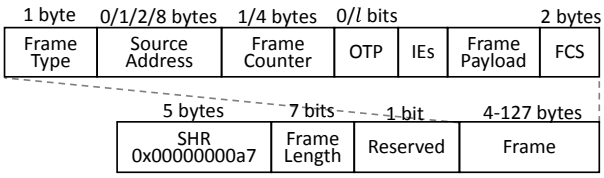

Figure 4: Format of 802.15.4 frames as adapted by POTR

C. On-the-fly Rejection of Unwanted 802.15.4 Frames

In the following, we explain the checks that POTR performs while receiving a frame for every single frame type. Throughout, if a check fails, POTR rejects the incoming frame by disabling the receive mode immediately.

1) Unicast Data Frames: Upon receipt of a unicast data frame, POTR first ensures that the sender is currently stored as a permanent neighbor by inspecting the Source Address field. If so, POTR generates the corresponding OTP and checks if it matches the one in the frame. The OTP of a unicast data frame is generated as

$$\text{KDF}(K_{\text{src},*} \oplus K_n, ID_{\text{dst}} \| C) \in \{0,1\}^l \qquad (1)$$

where KDF is a key derivation function, $\oplus$ denotes XOR, $K_{\text{src},*}$ is the group session key of the sender, K_n is the predistributed network-wide key, ID_{dst} is the receiver's address, $\|$ denotes concatenation, and C is either the sender's respective unicast frame counter in full when using the LB optimization or the sender's overall frame counter in full when not using the LB optimization. Finally, POTR makes sure that C was not replayed and, if so, updates the anti-replay data about the sender.

2) Broadcast Data Frames: As for broadcast data frames, POTR proceeds in the same way except that ID_{dst} is set to 0xff . . . ff and that C is either the sender's broadcast frame counter in full when using the LB optimization or the sender's overall frame counter in full when not using the LB optimization.

3) Acknowledgement Frames: Acknowledgement frames consist of the Frame Type field, the 8 least significant bits of the frame counter of the unicast frame whose receipt is being acknowledged, and the FCS field. To protect from denial-of-sleep attacks with acknowledgement frames, POTR simply ensures that their length is 4 bytes by inspecting the Frame Length field. Future work should authenticate acknowledgement frames by, e.g., adding OTPs or MICs to them.

4) `HELLO`s: The OTPs of `HELLO`s are generated like the ones of broadcast data frames, but POTR only rejects OTPs of `HELLO`s if they are replayed. In particular, if

Table I: Frame Types Supported by POTR

Scope	Frame Type	Description	Source Address	Frame Counter	OTP
unicast	unicast data	carries unicast data of upper layers (e.g. 6LoWPAN [29])	√	√	√
broadcast	broadcast data	carries broadcast data of upper layers (e.g. 6LoWPAN [29])	√	√	√
N/A	acknowledgement	part of ContikiMAC	×	√	×
broadcast	HELLO	part of AKES	√	√	√
unicast	HELLOACK	part of AKES	√	√	√
unicast	ACK	part of AKES	√	√	√
unicast	unicast command	unicasts processed by the link layer (e.g. UPDATEs and UPDATEACKs)	√	√	√
broadcast	broadcast command	broadcasts processed by the link layer (e.g. ANNOUNCEs [30])	√	√	√

a HELLO originates from a sender that is not currently stored as a permanent neighbor, the HELLO still needs to be processed to enable session key establishment with new neighbors. Also, even if a sender is stored as a permanent neighbor, it may send an invalid OTP because the sender may have rebooted and therefore generated a new group session key. To protect from denial-of-sleep attacks with HELLOs anyway, POTR employs four further checks. First, POTR ascertains that the length of incoming HELLOs is allowed. Second, POTR makes sure that there is space for storing another neighbor. Third, POTR ignores HELLOs from senders that are stored as tentative neighbors already. Forth, like AKES, POTR ignores HELLOs if the number of tentative neighbors exceeds a threshold [25].

5) HELLOACKs: Since session keys are not yet established when sending HELLOACKs, HELLOACKs contain special OTPs that are generated as

$$\text{KDF}(K_n, ID_{\text{src}} \| R_{\text{dst}}) \in \{0,1\}^l \tag{2}$$

where K_n is the predistributed network-wide key, ID_{src} is the sender's address, and R_{dst} is the 64-bit random number that AKES adds to the corresponding HELLO for different purposes anyway, as depicted in Figure 2. To prevent the replay of HELLOACK OTPs, POTR caches accepted OTPs of HELLOACKs and rejects HELLOACKs with cached OTPs. Once the cache is full, POTR rejects every HELLOACK right away. POTR clears the cache as soon as AKES broadcasts the next HELLO.

6) ACKs: OTPs of ACKs are generated like the ones of HELLOACKs, but using the 64-bit random number that AKES adds to the corresponding HELLOACK, as shown in Figure 2. To prevent the replay of ACK OTPs, POTR rejects ACKs from senders that are not stored as tentative neighbors. Furthermore, if POTR accepts an ACK while AKES rejects it, AKES also deletes the ACK sender from the list of tentative neighbors such that further ACKs from that sender are being ignored.

7) Unicast and Broadcast Command Frames: Finally, POTR treats unicast and broadcast command frames like unicast and broadcast data frames, respectively.

In this regard, we note that UPDATEs and UPDATEACKs, originally, contained unicast frame counters in full, irrespective of using the LB optimization or not. This allowed nodes to resynchronize each other's unicast frame counters. In our context, this would, however, defer the reception of the OTP field and hence the rejection of injected and replayed UPDATEs and UPDATEACKs. Therefore, we opted to add only the 8 least significant bits of the respective unicast frame counter to UPDATEs and UPDATEACKs when using the LB optimization. Thus, when using the LB optimization together with POTR, UPDATEs and UPDATEACKs only serve for (i) checking if a node is still in range and (ii) resynchronizing broadcast frame counters. Nevertheless, if a node loses track of a permanent neighbor's unicast frame counter, AKES will eventually delete that neighbor. Then, both nodes will start a new session as soon as any of them broadcasts a HELLO. After all, getting out of sync is very unlikely since a node has to miss $2^8 = 256$ unicast or broadcast frames in a row [25].

V. Security Analysis

In this section, we analyze four ways to bypass POTR's checks, namely local replay attacks, hidden wormholes, guessing attacks, and node compromises.

A. Local Replay Attacks

While attackers can use hidden wormholes to replay captured frames in another part of the victim network, attackers can also just capture and replay frames locally. However, POTR prevents the local replay of OTPs that are generated as per (1) by comparing frame counters, as described in Section IV-C1. Furthermore, OTPs that are generated as per (2) are protected against local replay attacks using special mechanisms, as explained in Section IV-C5 and IV-C6. On the other hand, if a node misses a frame, an attacker can inject a frame with the same frame header, but a different payload. Such a frame passes POTR's on-the-fly rejection once, but never the validation of MICs of 802.15.4 security (unless an attacker guesses a valid MIC). Thus, POTR does not obviate the need for adding MICs to frames.

B. Hidden Wormholes

As opposed to frame masking, POTR also resists hidden wormholes:

1) Indefinite broadcast frames: When using POTR, only HELLOs are destined to nodes that are not currently stored as neighbors. Replaying HELLOs elsewhere causes only a small energy consumption since the length of HELLOs is restricted. Additionally, POTR immediately

reject a HELLO (i) if its OTP was replayed, (ii) if there is no space for storing another neighbor, (iii) if the sender of the HELLO is already stored as a tentative neighbor, and (iv) if the number of tentative neighbors exceeds a threshold. These additional checks keep the energy consumption of multiple consecutive HELLOs low, as well.

2) Targeted broadcast frames: Replaying OTPs of other broadcast frames elsewhere will not wake up any node. On the one hand, this is because receivers ascertain that senders of broadcast data and command frames are stored as permanent neighbors. On the other hand, if an attacker spoofs the Source Address field, receivers will consider replayed OTPs of broadcast data and command frames as invalid since each node has its own group session key.

3) Unicast frames: Analogously, unicast data and command frames only wake up the intended receiver. Also, HELLOACKs and ACKs only wake up the intended receiver due to entangling the receiver's random number in their OTPs. Nevertheless, what an attacker can do is, instead of replaying single ACKs and HELLOACKs elsewhere, tunneling the whole three-way handshake of AKES between a pair of distant nodes, such that they become permanent neighbors [25]. In effect, the routing topology may get reorganized, which is very energy consuming, too [31]. Fortunately, the Routing Protocol for Low-Power and Lossy Networks (RPL) [32], which is also implemented by Contiki, at least mitigates such attacks by reorganizing the routing topology only after a hysteresis [33].

C. Guessing Attacks

Since OTPs have a length of l bits, the probability of guessing an OTP right is 2^{-l}. Of course, longer OTPs are harder to guess, while shorter OTPs incur less per frame overhead. Another advantage of shorter OTPs is that they reduce the time spent in interrupt service routines (ISRs), as we will discuss in Section VII-B. On the other hand, if an attacker manages to guess an OTP right, the respective anti-replay data is updated. As a consequence, legitimate frames may be considered as replayed and, in the worst case, nodes have to start a new session so as to resynchronize each other's unicast frame counters. But, even if an attacker guesses an OTP right, 802.15.4 security will eventually reject any injected frames, thereby rendering successful guessing attacks relatively harmless. We suggest setting $l = 24$ to trade off against all these aspects.

D. Node Compromises

Unfortunately, POTR does not resist node compromises. Once a group session key and the network-wide key K_n leaks, denial-of-sleep attacks become possible. Future work should address this vulnerability, but, for the scope of this paper, we assume that node compromises are an acceptable risk.

VI. Implementation

We implemented POTR with CC2538 transceivers and integrated POTR into the Contiki operating system. Presently, CC2538 transceivers are very popular and are, e.g., built into Zolertia Re-Motes and OpenMotes [34], both of which are prototyping platforms for IoT devices. Likewise, Contiki is a popular operating system for IoT devices.

CC2538 transceivers provide three interrupts for parsing incoming frames during receipt. First, the SFD interrupt issues when an SHR was detected. Second, the FIFOP interrupt issues as soon as FIFOP_THR bytes can be parsed or when a frame was completely received, whatever comes first. (The FIFOP interrupt has a different behavior when enabling the in-built frame filtering of the CC2538 transceiver. However, since the in-built frame filtering of CC2538 transceivers validates against the original 802.15.4 frame format and not POTR's frame format, our implementation disables the in-built frame filtering of CC2538 transceivers.) Third, the RXPKTDONE interrupt issues when a frame was completely received.

Our implementation configures the FIFOP interrupt to issue when the Frame Counter field is received. For this, we set FIFOP_THR to the appropriate value, which depends on the employed addressing mode and on whether the LB optimization is enabled. All this information is available at compilation time. Within the FIFOP ISR, POTR performs its checks. Many unwanted frames can be filtered out without waiting for the OTP field to arrive. Otherwise, if it is necessary to validate an OTP, we (i) generate the expected OTP, (ii) wait for the OTP field to arrive, and (iii) check if both OTPs match. Thus, the validation and reception of an OTP happens in parallel. As key derivation function, we use the hardware-accelerated Advanced Encryption Standard (AES) block cipher of the CC2538 transceiver and truncate its output to l bits. Finally, we use the RXPKTDONE ISR to send acknowledgement frames instantly.

The main difficulty with integrating POTR into Contiki was concurrency. This is because the bulk of Contiki's code can not be called from within an interrupt context. To resolve this problem, we added four locks, which protect code regions that should not be interrupted by POTR. For example, if another module currently uses the hardware-accelerated AES block cipher of the CC2538 transceiver, POTR will not call AES. Instead, if any of these locks is set as the FIFOP interrupt issues, we switch off the receive mode, i.e., ignore the incoming frame. In addition, we adapted Contiki's packetbuf module to support parsing incoming frames within an interrupt context.

VII. Evaluation

Using our implementation, we now demonstrate that, in comparison to using no denial-of-sleep defense, POTR reduces the time that 802.15.4 nodes spend in receive mode upon receiving an injected or replayed 802.15.4 frame by a factor of up to 16. Furthermore, we argue that the rejection speed of POTR is comparable with the rejection speed of the in-built frame filtering of CC2538 transceivers. Lastly, we determine that the processing and memory overhead of POTR is small.

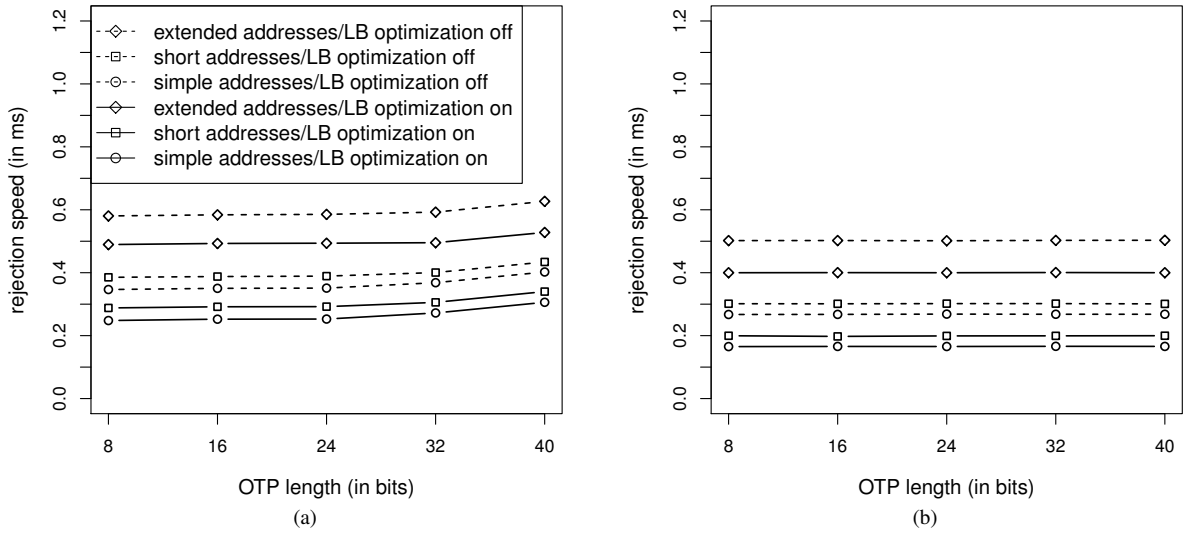

Figure 5: Rejection speed (a) when validating OTPs and (b) when skipping over the validation of OTPs (means over 1000 samples)

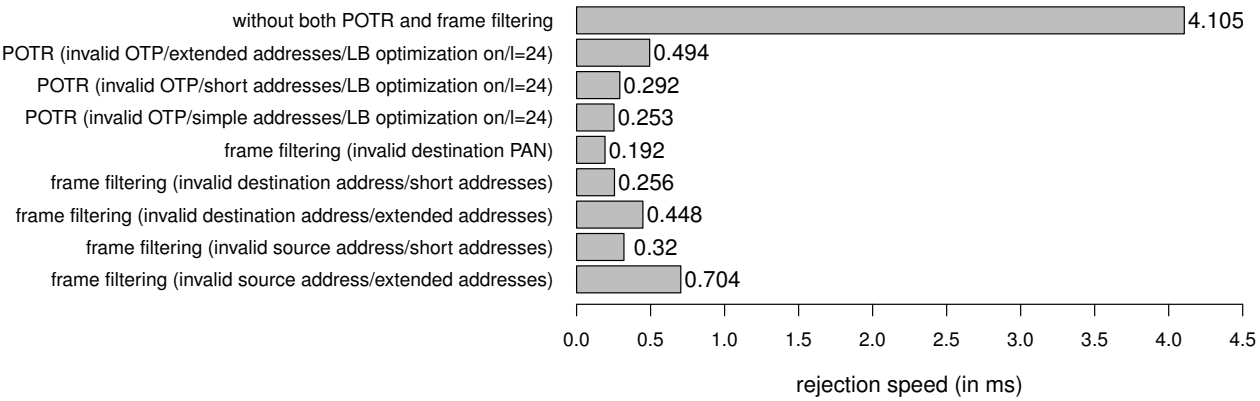

Figure 6: Comparison between the rejection speed of the in-built frame filtering of CC2538 transceivers and the rejection speed of POTR

A. *Rejection Speed*

As a baseline for comparison, an OpenMote A injected 127-byte frames. Another OpenMote B, which had both POTR and its in-built frame filtering disabled, received these frames and logged the time between `SFD` and `RXPKTDONE` interrupts. It turned out that the mean time between `SFD` and `RXPKTDONE` interrupts is 4.105ms. This value meets our expectations since, in the 2.4GHz band, 802.15.4 has a data rate of 250kbit/s. Furthermore, the Frame Length field has to be added to the 127 bytes and hence the time between `SFD` and `RXPKTDONE` interrupts should theoretically be $\frac{(127+1)\times 8\text{bit}}{250\text{kbit/s}} = 4.096\text{ms}$.

To measure the time until POTR rejects an injected or replayed 802.15.4 frame, an OpenMote A injected 127-byte unicast data frames, each of which contained the source address of a permanent neighbor of another OpenMote B, as well as an invalid OTP. The OpenMote B, which had POTR enabled, received these frames and logged the time between `SFD` interrupts and the moment when the frame reception aborts, which is signaled by `RXABO` interrupts. This experiment was repeated with and without the LB optimization, for simple, short, and extended addresses, as well as with the OTP lengths $l \in \{8, 16, 24, 32, 40\}$. Also, this experiment was repeated using frames that did not contain a source address of a permanent neighbor of B, such that B skipped over the validation (and reception) of OTPs.

Figure 5 shows how quick B rejects the injected 127-byte frames. The employed addressing mode has a significant effect on the rejection speed. This is because the `FIFOP` interrupt issues earlier or later, depending on the employed addressing mode. Thus, it is advisable to minimize the length of addresses. Likewise, using the LB optimization accelerates the rejection speed because the Frame Counter field gets 3 bytes shorter. For instance, as for simple addresses, with LB optimization, and $l = 24$, it takes 0.253ms to reject the injected 127-byte frames. This amounts to an improvement by a factor of $\frac{4.105}{0.253} = 16.23$. As shown in Figure 5b, B rejects frames even faster when skipping over the validation of OTPs, but attackers usually provoke the worst case.

Since POTR requires disabling the in-built frame filtering of CC2538 transceivers, a comparison with the rejection speed of the in-built frame filtering of CC2538 trans-

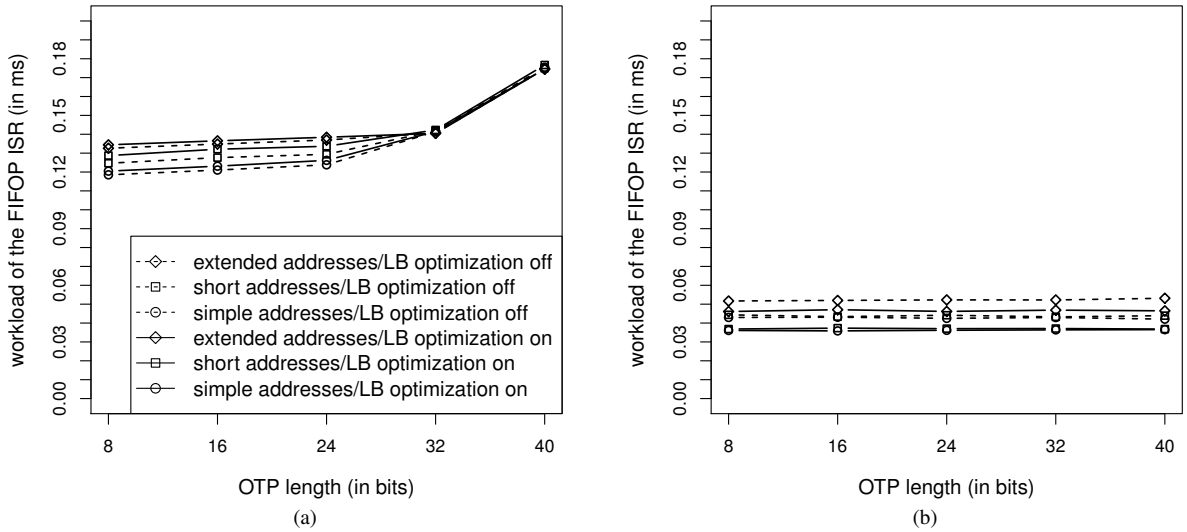

Figure 7: Workload of the FIFOP ISR (a) when validating OTPs and (b) when skipping over the validation of OTPs (means over 1000 samples)

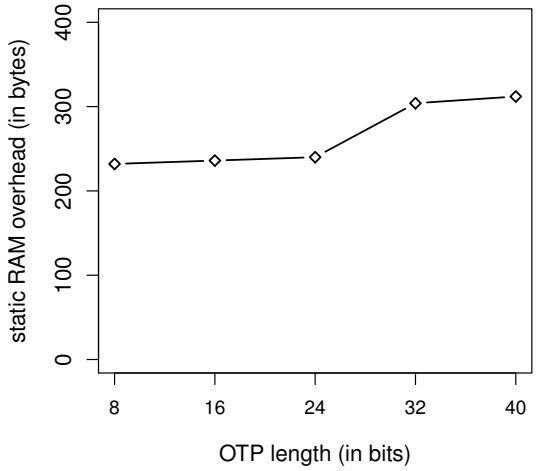

Figure 8: Static RAM overhead of POTR

ceivers is in order. The in-built frame filtering of CC2538 transceivers has three levels. First, the format validation of CC2538 transceivers validates incoming frames against the 802.15.4 frame format. Second, address recognition filters out incoming frames that are destined to other nodes by inspecting the Destination PAN and Destination Address fields. Third, source address matching (SAM) optionally checks if the source address of an incoming frame is contained in a list of acceptable source addresses.

In order to estimate the rejection speed of the in-built frame filtering of CC2538 transceivers, recall that, in the 2.4GHz band, 802.15.4 has a data rate of 250kbit/s. Furthermore, note that CC2538 transceivers do neither support suppressing the Sequence Number field nor simple addresses. Therefore, prior to rejecting a frame with an invalid PAN identifier, the Frame Length field, the Frame Control field, the Sequence Number field, as well as the Destination PAN Identifier field need to be received. This takes 0.192ms. Moreover, if a frame contains an invalid short or extended destination address, it takes additional 0.064ms or 0.256ms to reject that frame, respectively. SAM recognizes that a frame originates from an unwanted sender even later. If the source PAN identifier is suppressed, which is possible if it matches the destination PAN identifier, SAM rejects frames with unwanted source addresses only after 0.32ms and 0.704ms when using short and extended addresses, respectively.

Figure 6 compares the rejection speed of POTR with that of the in-built frame filtering of CC2538 transceivers. Both techniques constitute a huge improvement over rejecting unwanted 802.15.4 frames only after they were received since receiving a maximum-length 802.15.4 frame of 127 bytes takes 4.105ms. However, attackers can easily bypass the in-built frame filtering of CC2538 via spoofing, whereas bypassing POTR's checks is, according to our security analysis, impossible.

Another interesting observation from Figure 6 is that the rejection speed of the in-built frame filtering of CC2538 transceivers is comparable with the rejection speed of POTR. Further, since POTR rejects any unwanted frames during receipt, including frames that the in-built frame filtering of CC2538 transceivers would reject, there is no disadvantage in disabling the in-built frame filtering of CC2538 transceivers when using POTR.

B. Processing and Memory Overhead

To determine the processing overhead of POTR, the experiment of Figure 5 was repeated with the only difference being that B logged the time that POTR spends in the FIFOP ISR. Figure 7 shows the results. When validating OTPs (Figure 7a), the workload of the FIFOP ISR increases as l exceeds 24. This is because, at this point, the reception of OTPs takes longer than their validation, thus necessitating busy-waiting within the FIFOP ISR. When skipping over the validation of OTPs (Figure 7b), the FIFOP ISR just parses until the Frame Counter field

and concludes that the sender is not stored as a permanent neighbor. If a sender is not stored as a permanent neighbor, the `FIFOP` ISR also skips over restoring the LB-optimized frame counter, which is why the workload of the `FIFOP` ISR gets lower when enabling the LB optimization. A general observation is that shorter addresses reduce the workload of the `FIFOP` ISR, which can be attributed to the fact that less bytes are being processed.

The memory overhead of POTR was determined using the tool `arm-none-eabi-size`. It turned out that POTR's overhead in terms of program memory is 956 bytes and 820 bytes with and without the LB optimization, respectively. POTR's static RAM overhead is shown in Figure 8. The results in Figure 8 are irrespective of using the LB optimization or not. The jump at $l = 32$ comes down to an alignment change in a `struct`.

VIII. Conclusions and Future Work

Injected and replayed 802.15.4 frames expend the limited energy of 802.15.4 nodes. We have proposed POTR to reject injected and replayed 802.15.4 frames early during receipt. Overall, the processing and memory overhead of POTR is small. Beyond that, POTR incurs no communication overhead. Concerning rejection speed, POTR can not catch up with frame masking since frame masking avoids that 802.15.4 nodes detect injected and replayed 802.15.4 frames in the first place. On the other hand, frame masking can be bypassed and can not be implemented with CC2538 transceivers. Thus, POTR is a more effective and more practical solution. Unfortunately, broadcast and droplet attacks are by no means the only denial-of-sleep attacks on ContikiMAC. In fact, we suspect ContikiMAC's mechanism for learning the wake-up times of neighboring nodes to be vulnerable to pulse-delay attacks [35]. Moreover, by emitting noise, attackers can trick nodes into staying awake for extended periods of time [7]. For future work, we plan to devise countermeasures against these kinds of denial-of-sleep attacks on ContikiMAC, too.

References

[1] "IEEE Standard 802.15.4e," 2012.

[2] A. Dunkels, "The ContikiMAC radio duty cycling protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, 2011.

[3] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004)*. IEEE, 2004, pp. 455–462.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*. ACM, 2004, pp. 95–107.

[5] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. Le Roux, "WiseMAC, an ultra low power MAC protocol for the wiseNET wireless sensor network," in *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03)*. ACM, 2003, pp. 302–303.

[6] D. Moss and P. Levis, "BoX-MACs: exploiting physical and link layer boundaries in low-power networking," Stanford University, Tech. Rep. SING-08-00, 2008.

[7] M. Sha, G. Hackmann, and C. Lu, "Energy-efficient low power listening for wireless sensor networks in noisy environments," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN '13)*. ACM, 2013, pp. 277–288.

[8] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: scalable opportunistic routing with ORPL," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13)*. ACM, 2013, pp. 2:1–2:14.

[9] G. Z. Papadopoulos, A. Gallais, T. Noel, V. Kotsiou, and P. Chatzimisios, "Enhancing ContikiMAC for bursty traffic in mobile sensor networks," in *Proceedings of IEEE SENSORS 2014*. IEEE, 2014, pp. 257–260.

[10] B. Al Nahas, S. Duquennoy, V. Iyer, and T. Voigt, "Low-power listening goes multi-channel," in *Proceedings of the 2014 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2014, pp. 2–9.

[11] M. Michel, T. Voigt, L. Mottola, N. Tsiftes, and B. Quoitin, "Predictable MAC-level performance in low-power wireless under interference," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (EWSN '16)*. Junction, 2016, pp. 13–22.

[12] M. Brownfield, Y. Gupta, and N. Davis, "Wireless sensor network denial of sleep attack," in *Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop (IAW '05)*. IEEE, 2005, pp. 356–364.

[13] D. R. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of denial-of-sleep attacks on wireless sensor network MAC protocols," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, pp. 367–380, 2009.

[14] Z. He and T. Voigt, "Droplet: a new denial-of-service attack on low power wireless sensor networks," in *Proceedings of the 2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2013)*. IEEE, 2013, pp. 542–550.

[15] A. Wood, J. Stankovic, and G. Zhou, "DEEJAM: defeating energy-efficient jamming in IEEE 802.15.4-based wireless networks," in *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07)*. IEEE, 2007, pp. 60–69.

[16] *2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. C)*, Texas Instruments, http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[17] *CC2538 SoC for 2.4-GHz IEEE 802.15.4 & ZigBee/ZigBee IP Applications User's Guide (Rev. C)*, Texas Instruments, http://www.ti.com/lit/ug/swru319c/swru319c.pdf.

[18] R. Falk and H.-J. Hof, "Fighting insomnia: a secure wake-up scheme for wireless sensor networks," in *Proceedings of the Third International Conference on Emerging Security Information, Systems and Technologies (SECURWARE '09)*, 2009, pp. 191–196.

[19] S. Aljareh and A. Kavoukis, "Efficient time synchronized one-time password scheme to provide secure wake-up authentication on wireless sensor networks," *International Journal of Advanced Smart Sensor Network Systems (IJASSN)*, vol. 3, 2013.

[20] C.-T. Hsueh, C.-Y. Wen, and Y.-C. Ouyang, "A secure scheme against power exhausting attacks in hierarchical wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 6, pp. 3590–3602, 2015.

[21] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*. ACM, 2006, pp. 307–320.

[22] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. ACM, 2010, pp. 1–14.

[23] Q. Ren and Q. Liang, "Secure media access control (MAC) in wireless sensor networks: intrusion detections and countermeasures," in *Proceedings of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004)*. IEEE, 2004, pp. 3025–3029.

[24] D. R. Raymond and S. Midkiff, "Clustered adaptive rate limiting: defeating denial-of-sleep attacks in wireless sensor networks," in *Proceedings of the Military Communications Conference (MILCOM 2007)*. IEEE, 2007, pp. 1–7.

[25] K.-F. Krentz and Ch. Meinel, "Handling reboots and mobility in 802.15.4 security," in *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC '15)*. ACM, 2015, pp. 121–130.

[26] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a secure sensor network communication architecture," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*. ACM, 2007, pp. 479–488.

[27] M. G. Gouda, Y. ri Choi, and A. Arora, "Antireplay protocols for sensor networks," in *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, J. Wu, Ed. CRC, 2005, pp. 561–574.

[28] H. S. Chiu and K.-S. Lui, "DelPHI: wormhole detection mechanism for ad hoc wireless networks," in *Proceedings of the 1st International Symposium on Wireless Pervasive Computing*. IEEE, 2006, pp. 6–11.

[29] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, Internet Engineering Task Force, 2011, updates RFC 4944.

[30] K.-F. Krentz, H. Rafiee, and Ch. Meinel, "6LoWPAN security: adding compromise resilience to the 802.15.4 security sublayer," in *Proceedings of the International Workshop on Adaptive Security & Privacy Management for the Internet of Things (ASPI '13)*. ACM, 2013.

[31] K.-F. Krentz and G. Wunder, "6LoWPAN security: avoiding hidden wormholes using channel reciprocity," in *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices (TrustED '14)*. ACM, 2014, pp. 13–22.

[32] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, IETF, 2012.

[33] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson, "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)," RFC 7416, 2015.

[34] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "OpenMote: open-source prototyping platform for the industrial IoT." in *Ad Hoc Networks*, vol. 155. Springer, 2015, pp. 211–222.

[35] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava, "Secure time synchronization in sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 11, no. 4, pp. 23:1–23:35, 2008.