Counteracting Denial-of-Sleep Attacks in Wake-up-radio-based Sensing Systems

Angelo T. Capossele, Valerio Cervo, Chiara Petrioli and Dora Spenza
Computer Science Department, University of Rome "La Sapienza"
email: {capossele, cervo, petrioli, spenza}@di.uniroma1.it

***Abstract*—Wake-up-radio-based sensing systems make use of radio-triggering techniques and ultra-low power wake-up receivers (WuRs) to enable on-demand asynchronous network wake ups. Thanks to this, they have the potential to achieve low latency data collection at minimum energy cost, thus meeting the challenging lifetime and quality-of-service demands of emerging Internet of Things (IoT) and Wireless Sensor Networks (WSNs) applications. However, the fact that nodes can be remotely activated on-demand makes wake-up-radio-based networks vulnerable to energy exhausting attacks. In this paper, with a focus on practical implementation and validation, we present a full-fledged solution to counteract Denial-of-Sleep (DoS) attacks to wake-up-radio-based sensing systems. A core component of our proposed solution is a key exchange protocol based on Elliptic Curve Cryptography (the Fully Hashed MQV protocol), which we use in conjunction with implicit certificates.**

I. Introduction

Ultra-low power radio-triggering techniques are rapidly emerging as a key technology to enable long-lasting applications in the Internet of Things (IoT) and Wireless Sensor Networks (WSNs) domains. Nodes equipped with a wake-up receiver (WuR) use a dedicated ultra-low power hardware component to monitor the channel. Thanks to this, they can remain in a power-saving sleep state until they receive a WuR request by another node that demands their activation. Radio-triggering techniques break the inherent tradeoff between energy efficiency and data latency typical of duty-cycling-based networks [1], [2]. In addition, as on-demand wake ups virtually eliminates the need for idle listening, the use of a wake-up radio can provide significant energy saving to the network, especially in event-based applications.

However, WuR-based networks are vulnerable to Denial-of-Sleep (DoS) attacks. DoS attacks target the availability of network resources by reducing the time nodes are able to spend in sleep mode. In particular, an attacker could keep nodes of a wake-up-radio-based networks awake indefinitely by continuously sending WuR requests to them. If successful, this type of attacks can have devastating effect on the lifetime of the network, reducing it from decades or years to mere days. Denial-of-Sleep attacks are thus a significant threat for long-lasting wake-up-radio-based sensing systems.

Common countermeasures used in traditional WSNs are generally not good candidates for securing WuR-based networks. Indeed, while existing solutions can protect from DoS attacks WSN nodes that are already awake, to secure a wake-up-radio-based networks it is also necessary to prevent an attacker from waking up nodes. To this end, nodes must be able to distinguish between legitimate WuR request and malicious ones. However, the limited data rate of current wake-up radio prototypes hinders the adoption of authentication-based solutions that are successfully employed in traditional WSNs.

In this paper, we present *AntiDoS*, a practical framework to counteract sleep deprivation attacks in wake-up-radio-based sensing systems. Our proposed approach leverage on a simple yet effective idea: the WuR address of each node should be generated and updated in a pseudo-random fashion, based on key material known only by authorized peers. In this way, an attacker will not be able to wake up a node unless he knows a shared secret key used to generate valid WuR addresses. To manage the exchange of the secret key among legitimate nodes, a strong and secure Key Management Protocol (KMP) is needed. To be practical in large-scale deployments, the KMP has to adopt flexible and lightweight strategies that can support the dynamic nature of the IoT and of WSNs. A core component of AntiDoS is thus a robust, flexible and lightweight Key Management Protocol based on Public Key Cryptography. This KMP is complemented leveraging on the Fully Hashed MQV protocol [3], an authenticated key agreement scheme that provides key materials used to generate secure WuR addresses.

In this paper, we provide the following contributions:

- We propose AntiDoS, a security protocol to counteract Denial-of-Sleep attacks in wake-up-radio-based Wireless Sensor Networks;
- We present the first implementation of the Fully Hashed MQV protocol for resource-constrained devices;
- We perform extensive simulations to evaluate the effectiveness of the proposed AntiDoS protocol and to quantify its overhead;
- We experimentally assess the feasibility of our proposed approach and validate its energy consumption by providing an implementation on resource-constrained devices.

The remainder of the paper is organized as follows. We review related works in Section II. In Section III, we discuss the system and the threat models of Denial-of-Sleep attacks in wake-up-radio-based WSNs. Our proposed protocol, AntiDoS, is presented in Section IV. The security analysis of our scheme is discussed in Section V. Performance evaluation results are presented in Section VI. We perform an experimental validation and a practical assessment of the performance of AntiDoS in Section VII. Section VIII concludes the paper.

II. Related Work

A. Denial-of-Sleep Attacks on Wireless Sensor Networks

Sleep deprivation attacks were introduced by Stejano and Anderson [4]. This type of attacks target the power supply of a node, with the goal of significantly increasing its power consumption and reducing its lifetime. Raymond et al. explore the impacts of Denial-of-Service attacks against popular MAC protocols for Wireless Sensor Networks [5]. They show that an attacker sleeping $92 - 99\%$ of the time is able to keep victim nodes using popular MAC protocols such as S-MAC and T-MAC awake 100% of the time. Countermeasures to sleep deprivation attacks targeting the energy-saving mechanisms peculiar to traditional Wireless Sensor Networks have been widely investigated in the literature [6]. However, so far only a handful of works have studied security-related problems in wake-up-radio-based WSNs. Falk and Hof were the first to investigate the problem of counteracting DoS attacks in WuR-based Wireless Sensor Networks [7]. Their proposed approach is based on the general idea of storing a list of valid secret wake-up token reference values (WUTRV) at each node. Whenever a WuR request is received, the received wake-up token (WUT) is compared with those stored in the WUTRV to determine whether it is valid. To prevent replay attacks, WUTs are changed when a node is going to sleep. Though some generic mechanisms to establish and update WUTs and WUTRVs are discussed, no specific approach is implemented and evaluated in terms of introduced overhead. In addition, an ideal WuR platform is assumed. AntiDoS builds on their work, providing a concrete implementation of a protected wake-up scheme that addresses practical aspects such as secure and effective keys setup protocols, support for both unicast and multicast/broadcast communication and recovery mechanisms for possible de-synchronization of nodes in the network. Aljareh and Kavoukis describe in [8] a time-synchronized one-time-password scheme to provide secure wake-up authentication. Their proposed solution is similar to that of Falk and Hof, but it relies on the fact that wake-up tokens are re-generated periodically, rather than after being used. In particular, in their scheme time is divided into timeslots of fixed size. At the beginning of each time slot, each node re-compute WUTs and its own WUTRVs, which are only valid for the current timeslots (regardless of them being used). This approach relies on accurate network-wide synchronization to work correctly, which either requires additional hardware or introduces synchronization overhead. In addition, each node can receive only a single valid WuR request during each timeslot. Depending on the timeslots duration, the traffic load and the topology of the network, limiting WuR requests to one per timeslot may beset network performance. This introduces a tradeoff between timeslots duration and power consumption. To the best of our knowledge, we are the first to propose, implement and experimentally validate a full-fledged protocol for counteracting DoS attacks in Wireless Sensor Network with radio-triggering capabilities.

B. KMPs based on Public Key Cryptography

Public Key Cryptography (PKC) provides a more secure authentication mechanism when compared to symmetric based counterparts, which typically assume a pre-shared key (or a set of keys in the case of random key pre-distribution schemes) among network's node. In fact, when using symmetric schemes, if an attacker compromise a node (or a set of nodes in the case of random key pre-distribution), the security of the whole network would be compromised. PKC schemes allow two or more peers to agree on a common secret key by establishing a secure channel in an authenticated manner. Thus, if a node is compromised, only the secure channels of that node would be compromised. Although the research community has long been skeptical about the feasibility of PKC techniques over sensor devices, recent implementations have demonstrated PKC based on Elliptic Curve Cryptography (ECC) is feasible even over severely-constrained devices [9], [10], [11]. However, the majority of current implementations still handles key negotiation and peer authentication via large X.509 certificates [12]. Due to their size, transmission of X.509 certificates requires considerable bandwidth usage, and significantly affects both latency and energy consumption. For this reason, recent research efforts concentrate on alternative solutions that better suit the requirements of resource and energy constrained devices. One of the most promising directions is the adoption of *implicit certificates*, which bind the identity of a node and its public key within a single data structure, and which can certify the authenticity of such a relation without an explicit signature [13]. Thanks to the significant reduction of the certificate size, and thus of the bandwidth, latency, and energy needed to transmit them, implicit certificates can be considered a powerful technique for conceiving enhanced *KMP* schemes for IoT systems [12]. A recent approach is presented by Galindo et al. in [14]. Their proposed schemes are based on the original MQV protocol [15] on elliptic curves (ECMQV), and supports both simple certificates (e.g., public keys digitally signed by the sink) and implicit certificate (SC-ECMQV). However, as the proposed schemes only consider a two-way key agreement protocol, they lack of Perfect Forward Secrecy (PFS) and are vulnerable to group representation and unknown key share attacks [16].

III. System Model and Attack Description

A. Architecture of wake-up-radio-based nodes

Figure 1 shows the reference architecture of a device of a wake-up-radio-based sensing systems, which is based on a recent WuR prototype by Spenza et al. [1]. We assume each node is equipped with a main radio transceiver working in the 2.4GHz ISM band. WuR sequences are received and transmitted through an external expansion board, mounting an out-of-band wake-up receiver operating in the 868MHz ISM band and a dedicated WuR transmitter. The expansion board also includes an ultra-low power microcontroller (MCU) that performs addressing. The MCU generates an interrupt

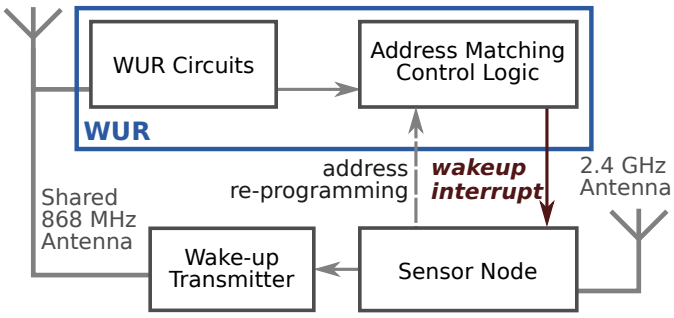

Fig. 1. The architecture of a WuR node.

whenever a valid WuR sequence is received, triggering the wake up of the node and the activation of its main radio. Using this architecture, the pool of WuR addresses assigned to the node can be dynamically managed and it is possible to assign multiple WuR addresses to the same node.

B. Threat model of DoS attacks in wake-up-radio-based WSNs

We model the attacker as a resource-rich device without energy constraints that can listen and transmit on both the WuR frequency and the main radio's channel.

Our proposed protocol addresses sleep deprivation attacks. In the considered threat model, the attacker aims at exhausting the energy reservoir of a wake-up-enabled device by awakening it when there is no legitimate network traffic, or by keeping it awake after a legitimate communication happens. In the first case, the attacker can try to guess a valid WuR address by bruteforce, or it can perform a reply attack. In particular, a replay attack is initiated by the attacker eavesdropping on the WuR frequency a valid WuR sequence sent to a target node. The intercepted WuR sequence is then periodically reinjected in the network, causing the victim node to awake and waste energy. If WuR addresses are static, e.g., as it is the case for ID-based wake-up protocols, the attacker can exhaust the entire energy reservoir of the target node by sending the same WuR sequence over and over. In case semantic WuR addressing is used, as in [1], [17], the WuR sequence sent by the attacker may become invalid as soon as the target node changes its own WuR address. However, even if the WuR address of the target is changed, the attack can be repeated by eavesdropping the next valid WuR sequence when a legitimate neighbor node awakes the target. This replay-based sleep deprivation attack has even more devastating effect if the attacker eavesdrops a broadcast WuR address. In this case, by sending a single WuR sequence, vast portions of network can be targeted at once. Finally, the attacker can also try to keep the target node awake after a legitimate communication takes place, by continuously sending on the main radio data packets destined to it.

IV. Anti-DoS Attacks Protocol

The objective of AntiDoS is that of securing the WuR interface against sleep deprivation attacks, by enabling successful wake ups only from authorized nodes. In particular, our proposed approach takes advantage of the specific features of the wake-up radio (Section III-A), which allow to dynamically manage the pool of wake-up addresses of each node.

Our proposed AntiDoS protocol is bootstrapped during an initial setup phase, during which the Key Management Protocol is run to establish a common secret key between legitimate peers using an authenticated key agreement protocol. The secret keys generated during the setup phase are then used during normal network operation to produce and update pseudo-random WuR sequences shared by the nodes in the network. In the following, we detail our proposed KMP and describe the address setup and communication mechanisms.

A. Key Management Protocol

The Key Management Protocol scheme employed in AntiDos is based on the exchange of four messages. The first two carry the key materials (i.e., the implicit certificate and a nonce), while the last two are used to ensure Perfect Forward Secrecy (PFS). The AntiDoS KMP makes use of implicit certificates, which offer joint authentication and key agreement services. In particular, with implicit certificates each node is able to compute, through the authenticated key agreement Fully Hashed MQV scheme (FHMQV), a shared secret key starting from an authenticated public key. The secret key negotiated during this procedure is then used to generate key materials for securing WuR addresses. In our implementation, we chose the Elliptic Curve based Que Vanston (ECQV) implicit certificate scheme [13], which produces short certificates of 160 bit. We now briefly review the generation of an ECQV implicit certificate. Unless otherwise specified, in the following we resort to multiplicative notation.

1) Generation of ECQV implicit certificates: ECQV defines an elliptic curve over a finite field $\mathbb{F}_q$ as $E/\mathbb{F}_q$, the cyclic subgroup $E(\mathbb{F}_q)$ as the set of points of this curve and $G \in \mathbb{G}_n$ as the generator of the cyclic subgroup of order n. To obtain an implicit certificate, a node A generates a random positive integer $k \in \mathbb{R}[1, n-1]$ and computes a point $R = G^k$ on the curve $E/\mathbb{F}_q$. Node A sends this point R to the sink, which acts as the Certification Authority (CA).

Upon reception of the point R, the CA generates a random positive integer $q \in \mathbb{R}[1, n-1]$ and computes the points $Q = G^q$ and $D = R * Q$. The CA then constructs the certificate data text, containing both identifying information and other information, such as the validity period of the implicit certificate. The implicit certificate P_A of node A is made by the public-key reconstruction data D and the certificate data text, in such a way that Q_A can be uniquely reconstructed. The CA signs the implicit certificate P_A, by first computing $h_{P_A} = \hat{H}(P_A)$, where $\hat{H}$ is a hash function with output of size n, and then calculating the implicit signature $s = \hat{H}(P_A)q + t \mod n$, where t is the CA's private key. The CA sends to node A both the implicit certificate P_A and the signature s. A can compute its private key as: $a = s + h_{P_A}k \mod n$. Network nodes can compute node A public key Q_A, provided they know the public key of the CA Q_{CA}

1) Node A sends a message with its implicit certificate, P_A, and an ephemeral secret exponent $\rho_A = G^{\rho_a}$, where $\rho_a \in R[1, n-1]$ and G is the generator of the elliptic curve of order n
2) Node B evaluates the implicit certificate P_A obtaining the public key Q_A of the remote device (as showed in equation 1), and does the following, as described in the FHMQV protocol:
   a) Compute an ephemeral secret exponent $\rho_B = G^{\rho_b}$
   b) Compute $d = \hat{H}(\rho_A, \rho_B, ID_A, ID_B)$
   c) Compute $e = \hat{H}(\rho_B, \rho_A, ID_A, ID_B)$
   d) Compute $s_B = \rho_b + eb \bmod n$, where b is node B's private key
   e) Compute $\sigma_B = (\rho_A Q_A^d)^{s_B}$
   f) Compute the Message Authentication Code (MAC) key $K_{MAC} = KDF_{MAC}(\sigma_B, ID_A, ID_B, \rho_A, \rho_B)$, where KDF_{MAC} is a Key Derivation Function with an output with the desired length of the MAC key K_{MAC}
   g) Compute $Auth_B = MAC_{K_{MAC}}(P_B, P_A, \rho_B, \rho_A)$
   h) Send a message to node A containing its implicit certificate P_B, its ephemeral secret exponent $\rho_B = G^{\rho_b}$ and a message with $Auth_B$
3) At reception of message from B, node A evaluates the implicit certificate P_B obtaining the public key Q_B of node B (as described in equation 1), and does the following:
   a) Compute $d = \hat{H}(\rho_A, \rho_B, ID_A, ID_B)$
   b) Compute $e = \hat{H}(\rho_B, \rho_A, ID_A, ID_B)$
   c) Compute $s_A = \rho_a + da \bmod n$, where a is node A's private key
   d) Compute $\sigma_A = (\rho_B Q_B^e)^{s_A}$
   e) Compute the MAC key $K_{MAC} = KDF_{MAC}(\sigma_A, ID_A, ID_B, \rho_A, \rho_B)$
   f) Verify that $Auth_B$ is valid
   g) Send $Auth_A = MAC_{K_{MAC}}(P_A, P_B, \rho_A, \rho_B))$ to node B
   h) Compute the shared key $K_{AB} = KDF(\sigma_A, ID_A, ID_B, \rho_A, \rho_B)$, where KDF is a Key Derivation Function with an output with the desired length of the shared key K_{AB}
4) At reception of the $Auth_A$ message, node B does the following:
   a) Verify that $Auth_A$ is valid
   b) Compute the shared key $K_{AB} = KDF(\sigma_B, ID_A, ID_B, \rho_A, \rho_B)$

Fig. 2. Fully Hashed MQV protocol

and the implicit certificate P_A which contains the public-key reconstruction data D and the identifying information of A:

$$Q_A = D^{h_{P_A}} Q_{CA} \tag{1}$$

Implicit certificates and the public key of the sink acting as the CA can be either preloaded in each device by the network administrator before deployment or they can be obtained directly from the sink after authentication.

2) Authenticated key exchange protocol: Figure 2 reports in detail the steps of the KMP used in AntiDoS. It is based on the Fully Hashed MQV protocol [3], an authenticated key agreement scheme based on the Full Exponential Challenge Response (FXRC) and Full Dual Exponential Challenge Response (FDCR) signature schemes. The FHMQV protocol is built upon a recent improvement of the original MQV protocol [15]. The steps reported in Fig. 2 assume that network node have already their respective implicit certificates. The shared keys generated during the key exchange phase are used to generated and update WuR addresses, as explained in Section IV-B. In addition, they are also used to cipher data packets exchanged by network nodes, as in [12].

B. Address Generation and Update

After the keys exchange phase, each node computes its own secure wake-up address pool. Let *A* and *B* be two neighbors that share a common secret key generated during the bootstrap phase (Section IV-A). Node *A* must compute its wake-up address so that it can be awakened by *B*. In particular, to communicate with *A*, node *B* needs to wake up node *A* by using the same wake-up address stored by *A*. Node *A* computes its WuR address as:

$$addr_{B \to A} = trunc(hash(K_{AB} || SN_{B \to A} || ID_B || ID_A)), \tag{2}$$

where K_{AB} is the shared key that *A* and *B* share, $SN_{B \to A}$ is a sequence number, ID_A and ID_B are the node IDs, *hash* is any cryptographic hash function (e.g., SHA256) and $||$ represents the concatenation operator. The sequence number $SN_{B \to A}$ is incremented after every interaction between *A* and *B*. This guarantees an attacker can not perform a replay attack by re-injecting in the network any WuR address he eavesdrops. The output of the hash function is truncated to the length of the wake-up address. The newly-computed WuR address is then stored by *A* in its WuR address pool. To communicate with *A*, node *B* computes the same WuR address $addr_{B \to A}$ using the shared key K_{AB}.

C. Unicast Communication

Any two nodes within communication range and sharing a common secret key can communicate by following the AntiDoS protocol shown by Fig. 3. When a node *A* needs to send a data packet to a node *B*, with which it shares a common secret key, it computes the wake-up address to wake up *B* as shown in Equation 2. The computed address is then sent to *B* using the WuR transmitter. Once awakened, *B* changes the state of its main radio from sleep to RX and waits to receive a

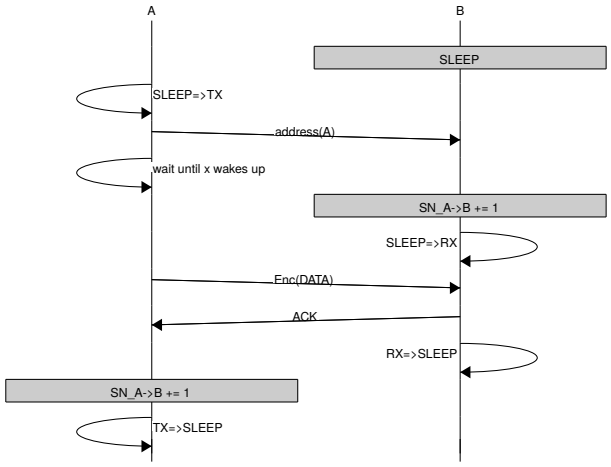

Fig. 3. Unicast communication: data transmission from *A* to *B*.

data packet from *A*. It also increments the sequence number of the link $A \rightarrow B$, recomputes its valid WuR address according to Eq. 2, removes from its addresses pool the WuR address it was just awaken with, and adds the new recomputed WuR address to its addresses pool. Once it receives the data packet, *B* sends back an ACK to *A* and then goes back to sleep. When *A* receives the ACK sent by *B*, it increments the sequence number $SN_{A \rightarrow B}$.

This procedure may fail in three cases: 1) *B* does not successfully awake due to interference or noise in the WuR channel; 2) the data packet from *A* to *B* is lost; or 3) the ACK from *B* to *A* is lost. In the first case, the two nodes remain synchronized, i.e., they still share the same sequence number $SN_{B \rightarrow A}$. To account for the possibility of *B* incorrectly receiving the WuR request, node *A* performs at most i attempts to wake up node *B*. If communication with *B* still fails after i attempts, *A* assumes the link with *B* is de-synchronized. A de-synchronization of the sequence number occurs whenever the data packet from *A* to *B* is lost or the ACK from *B* to *A* is lost. In both cases, *B* would increment the sequence number $SN_{A \rightarrow B}$, but *A* would not. This results in node *A* not being able to wake up node *B* any more. To recover from a de-synchronization, the sender node *A* tries to contact *B* using the next sequence number, i.e., $SN_{A \rightarrow B} + 1$. Indeed, *B* cannot be desynchronized by more than one sequence number with respect to *A*, as it can be awakened only once by using a valid sequence number. If *B* remains unreachable after j attempts of waking it up using the next sequence number, *A* re-starts the same protocol by performing another i attempts to wake up node *B*, using the current sequence number $SN_{A \rightarrow B}$. The sender repeats the synchronization recovery routine up to max_r times, after which the destination node is considered unreachable.

D. Multicast communication

The communication protocol described in Section IV-C enables secure wake ups for unicast communication. However, WSN network protocols usually relay on broadcast or multicast primitives to disseminate control information. Although it is possible to implement a secure multicast service using the secure unicast primitive detailed in Section IV-C, sending multiple unicast messages to simulate a multicast message is very energy costly. Using a shared multicast address for all the nodes in the network would be far more efficient in terms of energy consumption, but if an attacker intercepts it, the multicast address can be exploited to perform large-scale replay attacks. To address this limitation, AntiDoS offers a method to securely set up a multicast communication protocol that enables transmission of multicast messages at reduced energy costs.

The procedure consists in calculating, for each node, a multicast wake-up address shared with its neighbors, such that only the node itself is entitled to wake up all its neighbors with the transmission of a single WuR request. During this setup phase, each node *C* computes a random neighborhood secret and sends it to each neighbors belonging to its multicast tree using the unicast communication protocol. The secret is encrypted with a symmetric encryption algorithm (e.g., AES) using the common secret key between the sender node and its neighbor generated in the bootstrap phase (Section IV-A). Each node receiving the secret from *C* computes the multicast wake-up address of the *C*-neighborhood as:

$$mAddr_C = trunc(hash(nSecret_C || SN_C || ID_C)), \quad (3)$$

where $nSecret_C$ is the neighborhood secret provided by *C* and SN_C is the sequence number of the multicast link.

Node *C* can then send a multicast packet by computing the multicast WuR address as shown in Eq. 3, sending the WuR request to its neighbors, and then transmitting the multicast packet on the main radio's channel. Neighbors receiving the WuR request wake up to receive the multicast data packet, increment SN_C and remove the old address $mAddr_C$ from their WuR addresses pool. Once the data packet has been received, each neighbor node sends an ACK to *C* and goes back to sleep.

To avoid collisions between ACKs from multiple nodes, ACKs are sent according to a TDMA schedule. When a source node transmits a multicast data packet, it includes an enumeration of the timeslots assigned to each neighbor (in some arbitrary order), in which each slot is long enough for an ACK transmission. After the reception of the multicast data packet, each neighbor node goes back to sleep until it is its turn to transmit the ACK. Node *C* increments SN_C and goes back to sleep once all ACKs are received.

As in the case of unicast transmissions, it may happen that a node becomes desynchronized with respect to its neighborhood's multicast address. In particular, if a portion of the secure neighborhood is not awakened by the transmission of the multicast WuR request, a subset of neighbors might remain synchronized, while the rest of them is not. In this case, a mechanism similar to that adopted in the unicast communication protocol for recovering from a desynchronization state is employed. In particular, the sender node repeats the transmission up to m_i times, transmitting the

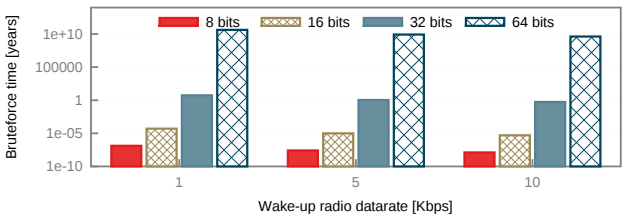

Fig. 4. Time needed for a brute-force attack when using as valid WuR address a single sequence of length 8, 16, 32 or 64 bits. The datarate of the WuR is set to 1Kbps.

multicast WuR request that contains the previous sequence number, in order to wake up only the nodes that were not awaken by the previous transmission. If not all neighbors recovers synchronization after this process, node *C* resurrects to using the unicast recovery procedure with each individual de-synchronized neighbor. In addition, in application scenarios where multicast packets are sent at periodic intervals, each node *B* can locally detect a de-synchronization occurred with respect to another node *C* if it does not receive any packet from *C* for a time longer than the periodic interval. In this case, *B* can send a resynchronization request to *C* through a unicast message. *B* authenticates the request along with a timestamp to avoid replay attacks, using the common key shared with *C*. Upon reception of a resynchronization request, *C* verifies that the request is legitimate and sends back the actual sequence number (possibly piggybacked in the ACK response), encrypted with the key shared with *B*.

V. AntiDoS Security Analysis

1) Brute-force Attack: As the key exchange phase detailed in SectionIV-A guarantees the attacker has no knowledge of any shared secret key, he cannot generate valid wake-up sequences. However, it can carry on a brute-force attack to guess a valid address. The time needed to perform such an attack depends on the length of wake-up sequences (including any WuR preamble), on the number of WuR addresses valid for a node (i.e., the size of its WuR pool) and on the data rate used by the WuR. In fact, even if we assume the attacker has unlimited computational and energy capabilities, the wake-up radio successfully recognizes only WuR sequences sent at a specific rate. Figure 4 shows the time needed for a brute-force attack for a single WuR sequence of length 8, 16, 32 or 64 bits, assuming the WuR datarate is 1 Kbps. Using a 1Kbps WuR and 32-bit sequences, the average time for a brute-force attack is of more than 2 years. Even if the attacker succeeds in guessing a valid sequence, it is only able to wake up the node once, as the node just awaken will return to sleep after a short amount of time (Section IV-C). In addition, the attacker would not have any information about the outcome of the attack, as the node will send an ACK only after a valid data transmission, which requires upper-layer authentication.

2) Reply Attack: The attacker can perform a reply attack by intercepting WuR requests. If a valid WuR request sent by a node does not reach the target node (i.e., it is lost due to noise), the attacker can successfully wake up the target node by resending it, as the sequence number of the target would not has been incremented. However, as in the previous case, the attacker would be only able to wake up the target node once. Although this kind of attack de-synchronizes the sender and the receiver node, synchronization is subsequently restored by using the technique presented in Section IV-C.

3) Keeping a node awake: The attacker could try to keep awake a node that has been awaken by a legitimate WuR sequence. However, since nodes go back to sleep after a pre-defined amount of time, such an attack would not be successful. Even if the attacker tries to keep a node awake by sending data packets to it, MAC-layer authentication would fail for packets generated by the attacker. The target node would thus go back to sleep immediately after the failed authentication.

4) Security of AntiDoS KMP: The use of implicit certificates with the FHMQV protocol in AntiDoS allows network nodes to authenticate to each other and to build a secret key, while providing efficiency and resiliency to ephemeral secret exponent leakage [16]. Moreover, implicit certificates, binding a public key to its owner in a trusted way, make the proposed strategy robust against Man In The Middle attacks [18]. The security of FHMQV relays on the elliptic curve discrete logarithm problem (ECDLP), whose hardness is the fundamental assumption of the security of various cryptographic protocols, such as the Elliptic Curve Diffie-Hellman (ECDH) and the Elliptic Curve Digital Signature Algorithm (ECDSA). It is possible to demonstrate that the mutual authentication scheme implemented in the second part of the protocol protects the entire approach against replay attacks [16]. The latest two messages $Auth_A$ and $Auth_B$ in Fig. 2 offer the same functions of the *Finished* message in the Transport Layer Security (TLS) protocol [19]. Similarly to TLS, the latest two messages of the proposed KMP carry the authenticated information that is computed by considering all the messages, including ephemeral secret exponent values, exchanged before. As a consequence, the validity of the adopted mutual authentication scheme relies on the same security proof of the TLS protocol [20]. Moreover, the computation of the ephemeral secret exponents ρ_A and ρ_B (Fig. 2) can be performed during idle time (e.g., by exploiting pre-computation techniques such as IBPV [21]). Thus, side-channel attacks can not obtain any information about the computation time needed for computing a secret exponent.

VI. Performance Evaluation

We implemented AntiDoS in GreenCastalia, an open-source extension to the Castalia simulator [22] that we have developed to accurately model energy-related aspects of WSNs [23]. We consider networks deployed for environmental monitoring, where 100 nodes are randomly and uniformly distributed over an area of 200m$\times$200m. Data traffic is generated according to a Poisson process of intensity λ packets per second. Once generated, each data packet is assigned to a source node chosen randomly and uniformly among the network nodes, except

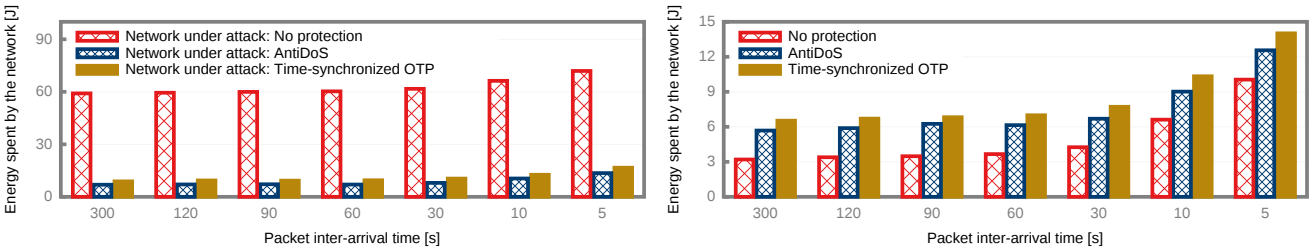

Fig. 5. Energy consumption of the baseline solution, of AntiDoS and of the time-synchronized one-time-password scheme in two scenarios: (a) Network under attack; (b) No attacker.

the neighbors of the sink that only act as relays. Source nodes, which are equipped with on-board Sensirion SHT1x sensors, perform temperature and humidity measurements and generate a data packet to report sensor readings to the sink. Based on the sensor datasheet specifications, we set the power consumption of sensing to 3mW, and the time needed for the measurement to complete to 171ms [24]. The size of each data packet is set to 35B to include application payload (sensing measurements) and headers and trailers added by the lower layers. The channel data rate is 250Kbps. Routing in the network is performed by using the Collection Tree Protocol (CTP) [25]. We implemented CTP in GreenCastalia based on the CTP implementation publicly available at [26], which we extended to support wake-up based networks [27]. In our simulations, we use the energy model of the MagoNode++, a mote platform we recently proposed that integrates wake-up-radio capabilities [28]. Our developed energy model also account for the power consumption of both the WuR (including its integrated MCU) and the WuR transmitter (the low-power CC1101 transceiver from Texas Instruments, transmitting WuR addresses at $+10$dBm). Simulations take into account the power consumption of the node in different states, as well as the transition times and the average power consumption of each of the node components during state switching. Mimicking the TinyOS implementation, the MCU of the nodes is put into low-power sleep state whenever the radio is off and the node is not performing other tasks [29]. Energy consumption of security-related operations is modeled according to experimental measurements (see Section VII). The default GreenCastalia settings are used for channel and radio models. The transmission power of the main transceiver is set to -3dBm for energy conservation, and its transmission range is 70m. The average path loss between nodes in the network is estimated by using the lognormal shadowing model. Packet collisions are determined based on the additive interference model. WuR sequences are sent at 1 Kbps, and consists of 4B of data preceded by 2 bits of preamble. The WuR is modeled according to experimental measurements and specifications of a recent WuR prototype (Section III-A). In particular, the power consumption of the WuR is set to 1.276μW, while its sensitivity is of -55dBm. The wake-up probability is modeled as a function of the power received by the WuR and of the data rate used to transmit wake-up sequences.

All results have been obtained by averaging the outcomes of a number of simulation runs large enough to obtain a 95% confidence interval and 5% precision. Each run lasts 3600s. In order to evaluate steady-state performance, all metrics are collected after the initial network setup phase.

We evaluate the performance of three different protocols:

1) The baseline solution, i.e., our implementation of CTP for wake-up-radio-based networks without any protection against sleep deprivation attacks;
2) CTP with the AntiDoS protocol with the FHMQV-based key exchange protocol with implicit certificates;
3) The time-synchronized one-time-password scheme [8], using the Tiny Encryption Algorithm (TEA) [30] for tokens encryption and the Flooding Time Synchronization Protocol (FTSP) [31] for time synchronization.

Performance of each protocol is evaluated in two different scenarios: with and without the presence of an attacker. We model the attacker as follows. We assume the attacker has an architecture similar to that of a wake-up-enabled node (Section III-A), which includes a main radio, a wake-up receiver and a wake-up transmitter. Being not subject to energy constraints, the wake-up radio of the attacker has a higher sensitivity than that of a regular node and can transmit data packets and WuR requests using higher transmission power than a regular node. At the beginning of each simulations, the attacker is randomly placed in the network. Since a bruteforce attack is hard due to the limited data rate of the WuR and of the size of the address space (Section V), the attacker performs a replay attack. In particular, the attacker activates its wake-up radio and records the last N_w addresses it was able to overhear. It then re-broadcast the received sequences with a period p, which we set to 10 seconds in our simulations. We evaluate the impact of the reply attack on the performance of the network in Fig. 5. To quantify the overhead introduced by AntiDoS and by the time-synchronized one-time-password scheme, we also perform simulations in the scenario in which the network is not under attack.

Energy consumption results in both scenarios are shown in Fig. 5, which reports the total energy consumed by the network (but the sink). Reported results include the energy consump-

tion of the baseline solution, of the AntiDoS protocol and of the time-synchronized one-time-password scheme, with and without the presence of an attacker. We vary the data packet interarrival time between 300 and 5 seconds. As shown by the figure, the reply attack is very effective in the scenario in which the network is not protected: The total energy consumption of the network is increased by an order of magnitude by a single attacker performing a reply attack every 10 seconds. For example, when the data packet interarrival time is of 120s, simulation results (not reported here) show that the expected lifetime of the network (defined as the time when the first node dies) is reduced from over 26 years to less than one year. The attack is even more effective if the period of which WuR sequences are re-broadcasted is smaller, or if multiple attackers are present in the network. Fig. 5 also shows the effectiveness of AntiDoS against replay attacks: In presence of an attacker, the total energy consumed by AntiDoS is only slightly higher than that consumed in the scenario in which the network is not under attack. It is worth noting that the increase of energy consumption in presence of an attacker is *not* due to nodes in the network being successfully awaken by the attacker. Rather, it is caused by the fact that WuR sequences injected by the attacker cause interference with legitimate WuR requests, thus increasing the number of re-transmissions.

We then quantify the overhead introduced by AntiDoS and by the time-synchronized one-time-password scheme when the network is not under attack. As reported by Fig. 5b, the time-synchronized one-time-password scheme consumes around 14% more energy than AntiDoS, due to the additional overhead introduced by the TEA encryption and the FTSP protocol used for time synchronization.

VII. Experimental Validation

To demonstrate the viability of our proposed AntiDoS protocol on resource-constrained devices, we assess both the computational overhead of its most frequently-used operations, defined in terms of the time needed to perform them, and their energy consumption. To this end, we implemented AntiDoS in TinyOS [32]. Table I shows the computational overhead and energy cost of the main operations performed by AntiDoS over a MagoNode++. For comparison, we note that the encryption of a block size of 64 bit (the fixed block size for TEA) using the TEA algorithm employed in [8] takes 19.6ms on a MagoNode++ when using the recommended value of 64 Feistel rounds.

We recall we are using the multiplicative notation. We have implemented the ECC primitives which AntiDoS relays upon based on a NIST recommended Koblitz curve (sect163k1) defined over $\mathbb{F}_{2^m}$:

$$E(\mathbb{F}_{2^m}) : y^2 + xy = x^3 + ax^2 + b, \tag{4}$$

where $m = 163$ and the representation of $\mathbb{F}_{2^{163}}$ is defined by:

$$f(x) = x^{163} + x^7 + x^6 + x^3 + 1. \tag{5}$$

The NIST irreducible polynomial for the finite field $\mathbb{F}_{2^{163}}$ allows us to leverage on some optimizations, such as a fast modular reduction algorithm and Solinas' τ-radic nonadjacent form (TNAF) representation [33], for which efficient algorithms for field multiplication and inversion are available. In order to speed up the computation of the ephemeral secret exponents, which we recall it is in the form of G^x, where G is the generator of the elliptic curve $E(\mathbb{F}_{2^{163}})$ and x is the secret exponent, we used a technique similar to that proposed in [21], referred to as *IBPV*. Differently from the original *IBPV*, our version leverages on an expansion method based on the use of the Frobenius map τ, to generate pairs in the form k, G^k starting from a set of precomputed pairs. Results from Table I show that an ephemeral secret exponentiation operation requires 26 ms on the MagoNode++ platform. This optimization allows AntiDoS to generate ephemeral exponents with a reduction of a factor 6 in terms of time and energy w.r.t. the EC point exponentiation primitive. We have implemented the Message Authentication Code as a Hash-based message authentication code, specifically the HMAC-SHA256, which computes a HMAC in 20ms. The KDF has been implemented leveraging on the SHA-256 cryptographic hash function. In order to optimize the total number of operations, AntiDoS performs only one SHA-256 iteration to generate both the message authentication code key K_{MAC} and the shared key K_{AB}, thus $K_{MAC} = K_{AB}$. Following a similar approach, we perform the hash computation of d and e during the FHMQV protocol (Figure 2) as a single operation, by using a hash function with a longer output, specifically SHA384. The resulting output is then cut in half to provide d as $\hat{H}_{0,159}(\rho_A, \rho_B, ID_A, ID_B)$ and e as $\hat{H}_{160,319}(\rho_A, \rho_B, ID_A, ID_B)$, thus saving one hash computation. Overall, AntiDoS KMP takes 584 ms and consumes 8.23 mJ to perform a complete key exchange during the bootstrap phase, while it takes 10 ms to set up each WuR address using SHA-256 or 3 ms using SHA-160. As a comparison, the key exchange protocol using implicit certificates (SC-ECMQV) presented in [14] takes 896 ms at each node to establish a secure channel between two nodes, which would cost 12.63 mJ of energy on a MagoNode++. When using simple certificates digitally signed by the sink, the same operation takes 1236 ms due to the additional verification process needed to authenticate the public keys, which results in a energy consumption of 17.42 mJ.

These results show that our protocol design choices and implementation optimizations achieve the best performance in terms of overall energy consumption and computational

TABLE I
Computational overhead and energy consumption of AntiDoS operations on a MagoNode++.

Operation	Time	Energy Consumption
Scalar addition or multiplication	1 ms	14 μJ
EC point multiplication	2 ms	28 μJ
EC point exponentiation	170 ms	2.4 mJ
Ephemeral exponent generation	26 ms	0.37 mJ
SHA-160	3 ms	0.04 mJ
SHA-256	10 ms	0.14 mJ
SHA-384	15 ms	0.21 mJ
HMAC	20 ms	0.28 mJ

overhead, while also providing a higher security level due to provided perfect forward secrecy.

VIII. Conclusion

As applications of the IoT continue to grow in scope and complexity, security-related aspects became increasingly critical. Acknowledging the need for secure communication in last-generation sensing systems, in this paper we propose AntiDoS, a protocol to counteract Denial-of-Sleep attacks in wake-up-radio-based networks. AntiDoS can be combined with existing IoT standard protocols, such as 802.15.4 and 6LoWPAN, to augment the security mechanisms they provide by implementing protection against denial of sleep attacks targeting devices equipped with wake-up radios. AntiDoS takes advantage of the specific features of the wake-up radio we considered, which allows to dynamically manage the pool of wake-up addresses of each node. A core component of the AntiDoS protocol is a Key Management Protocol based on both implicit certificates and the Fully Hashed MQV scheme, which provides flexible and secure peers authentication and keys exchange. Results show that the AntiDoS protocol successfully counteracts Denial-of-Sleep attacks in wake-up-radio-based networks, while outperforming competing security schemes in terms of both computation and communication overhead. Finally, an implementation of our proposed solution over a recently-proposed wireless mote platform has confirmed the feasibility of our approach on resource-constrained devices.

Acknowledgment

The research contribution presented in this paper has been supported by the projects *Smartour SCN 00166 (Intelligent platform for tourism)* and *Social Museum & Smart Tourism* (MIUR CTN01_00034_23154) founded by MIUR.

References

[1] D. Spenza, M. Magno, S. Basagni, L. Benini, M. Paoli, and C. Petrioli, "Beyond Duty Cycling: Wake-up Radio with Selective Awakenings for Long-lived Wireless Sensing Systems," in *Proceedings of IEEE INFOCOM 2015*, Hong Kong, April 2015, pp. 522–530.

[2] F. Sutton, B. Buchli, J. Beutel, and L. Thiele, "Zippy: On-Demand Network Flooding," in *Proceedings of ACM SenSys 2015*, 2015, pp. 45–58.

[3] A. P. Sarr, P. Elbaz-Vincent, and J.-C. Bajard, "A secure and efficient authenticated Diffie-Hellman protocol," in *Public Key Infrastructures, Services and Applications*. Springer, 2010, pp. 83–98.

[4] F. Stajano and R. Anderson, "The Resurrecting Duckling: security issues for ubiquitous computing," *IEEE Computer*, vol. 35, no. 4, pp. 22–26, Apr 2002.

[5] 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, Jan 2009.

[6] D. R. Raymond and S. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 74–81, Jan 2008.

[7] R. Falk and H.-J. Hof, "Fighting Insomnia: A Secure Wake-Up Scheme for Wireless Sensor Networks," in *Proceedings of SECURWARE 2009*, June 2009, pp. 191–196.

[8] S. Aljareh and A. Kavoukis, "Efficient Time Synchronized One-Time Password Scheme to Provide Secure Wake-Up Authentication Wireless Sensor Networks," *International Journal Of Advanced Smart Sensor Network Systems*, vol. 3, January 2013.

[9] A. Capossele, V. Cervo, G. D. Cicco, and C. Petrioli, "Security as a CoAP resource: an optimized DTLS implementation for the IoT," in *Proceedings of IEEE ICC 2015*, London, UK, June 2015.

[10] S. Basu and M. Pushpalatha, "Analysis of energy efficient ECC and TinySec based security schemes in Wireless Sensor Networks," in *Proceedings of IEEE ANTS 2013*, Dec. 2013.

[11] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proceedings of IEEE SECON 2004*, Oct 2004, pp. 71–80.

[12] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia, and G. Bianchi, "Key management protocol with implicit certificates for IoT systems," in *Proceedings of ACM IoT-Sys 2015*, 2015, pp. 37–42.

[13] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[14] D. Galindo, R. Roman, and J. Lopez, "On the energy cost of authenticated key agreement in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 1, pp. 133–143, 2012.

[15] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 119–134, 2003.

[16] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in *Advances in cryptology–CRYPTO 2005*. Springer, 2005, pp. 546–566.

[17] C. Petrioli, D. Spenza, P. Tommasino, and A. Trifiletti, "A Novel wake-up Receiver with Addressing Capability for Wireless Sensor Nodes," in *Proceedings of IEEE DCoSS 2014*, Marina Del Rey, USA, 5 2014, pp. 18–25.

[18] D. R. L. Brown, R. P. Gallant, and S. A. Vanstone, "Provably Secure Implicit Certificate Schemes," in *Proceedings of the International Conference on Financial Cryptography*. Springer-Verlag, 2002.

[19] Dierks, T. and Rescorla, E., *The Transport Layer Security Protocol Version 1.1*, IETF, Apr. 2006.

[20] A. K. Ranjan, V. Kumar, and M. Hussain, "Security analysis of TLS authentication," in *Proceedings IC3I 2014*, Nov. 2014.

[21] G. Ateniese, G. Bianchi, A. Capossele, and C. Petrioli, "Low-cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Precomputation Techniques?" in *Proceedings of NDSS 2013*, San Diego, CA, February 24-27 2013.

[22] A. Boulis, "Castalia: Revealing Pitfalls in Designing Distributed Algorithms in WSN," in *Proceedings of ACM SenSys 2007*, Sydney, Australia, November 6–9 2007, pp. 407–408.

[23] D. Benedetti, C. Petrioli, and D. Spenza, "GreenCastalia: An Energy-harvesting-enabled Framework for the Castalia Simulator," in *Proceedings of ACM ENSSys 2013*, 2013, pp. 7:1–7:6.

[24] Sensirion AG, "SHT1x Datasheet: Humidity and Temperature Sensor IC," 2011.

[25] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An Efficient, Robust, and Reliable Collection Tree Protocol for Wireless Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, pp. 16:1–16:49, Dec. 2013.

[26] U. Colesanti and S. Santini, "The Collection Tree Protocol for the Castalia wireless sensor networks simulator," Department of Computer Science, ETH Zurich, Tech. Rep. 729, June 2011.

[27] S. Basagni, C. Petrioli, and D. Spenza, "CTP-WUR: The Collection Tree Protocol in Wake-up Radio WSNs for Critical Applications," in *Proceedings of IEEE ICNC 2016*, Kauai, Hawaii, Feb 2016.

[28] M. Paoli, D. Spenza, C. Petrioli, M. Magno, and L. Benini, "MagoNode++: A Wake-Up-Radio-Enabled Wireless Sensor Mote for Energy-Neutral Applications," in *Proceedings of ACM/IEEE IPSN 2016 (Poster Session)*, Vienna, Austria, April 2016.

[29] R. Szewczyk, P. Levis, M. Turon, L. Nachman, P. Buonadonna, and V. Handziski, "TEP 112: Microcontroller Power Management," 2006.

[30] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," in *Fast Software Encryption*. Springer, 1994, pp. 363–366.

[31] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The Flooding Time Synchronization Protocol," in *Proceedings of ACM SenSys 2004*, 2004, pp. 39–49.

[32] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "TinyOS: An operating system for sensor networks," *Ambient intelligence*, vol. 35, 2004.

[33] J. A. Solinas, "Efficient arithmetic on koblitz curves," *Des. Codes Cryptography*, vol. 19, no. 2-3, pp. 195–249, Mar. 2000.