SWARD: A Secure WAke-up RaDio against Denial-of-Service on IoT devices

Maxime Montoya, Simone Bacles-Min, Anca Molnos, Jacques J.A. Fournier

SWARD: A Secure WAke-up RaDio against Denial-of-Service on IoT devices

Maxime Montoya
Univ. Grenoble Alpes, CEA, LETI, DACLE
Grenoble, France
maxime.montoya@cea.fr

Simone Bacles-Min
Univ. Grenoble Alpes, CEA, LETI, DACLE
Grenoble, France
simone.bacles-min@cea.fr

Anca Molnos
Univ. Grenoble Alpes, CEA, LETI, DACLE
Grenoble, France
anca.molnos@cea.fr

Jacques J.A. Fournier
Univ. Grenoble Alpes, CEA, LETI, DSYS
Grenoble, France
jacques.fournier@cea.fr

ABSTRACT

Wake-up radios are mechanisms that control the sleep and active modes of energy-constrained Internet of Things (IoT) nodes. These radios detect pre-determined wake-up tokens and switch the devices to an active state. Such systems are vulnerable to a kind of Denial-of-Service attacks called Denial-of-Sleep, where attackers continuously send wake-up tokens to deplete the battery of the nodes. We propose a protocol to mitigate these attacks that includes a novel solution to generate hard-to-guess wake-up tokens at every wake-up. Simulations show that under standard operating conditions, it has a negligible energy overhead (0.03%), while it increases the lifetime of an IoT node by more than 40 times under Denial-of-Sleep attack. Finally, we compare our protocol to related work against Denial-of-Sleep attacks, and explain why it is both more resilient and more energy-efficient than existing approaches.

KEYWORDS

Denial-of-Service, Wake-up radio, Network security, Internet of Things, Wireless Sensor Networks

1 INTRODUCTION

Energy consumption is critical for battery-powered devices in the Internet of Things (IoT) and Wireless Sensor Networks (WSN), which have to be operational during a long time. These devices, or nodes, are usually idle most of the time, and can be switched to a low-power sleep state during this idle time, with the radio and most of the processing capabilities being disabled. Wake-up radios are typical mechanisms to control the sleep and active states of these nodes [8]. This solution consists of using an ultra-low power receiver with a low data rate, in addition to the main transceiver, that continuously monitors a dedicated communication channel in order to detect wake-up tokens sent by other nodes. When a wake-up token is received by this low-power receiver on the target node, it is compared to a reference token previously stored in this receiver; if they are identical, the node switches to active state. This wake-up token is usually a predefined code, identical for every wake-up, such as the address of the target node [8].

An attacker can easily obtain this token by listening to the communication channel, and replay it many times to the target node to wake it up continuously. This would rapidly deplete the battery of the node, and reduce its lifetime. This attack is called a Denial-of-Sleep attack [4]. In order to prevent Denial-of-Sleep, wake-up tokens have to be different for every wake-up. Moreover, they have to be unpredictable: an attacker able to retrieve one or several tokens should not be able to guess any of the following tokens.

Some methods have been proposed to generate wake-up tokens [4, 7, 9, 11, 12]. They either require additional communication on the main radio, or complex computation on data that has to be stored and sometimes continually updated between two wake-ups, which is energy-consuming. Most of these mechanisms use extra secret symmetric keys, dedicated to the wake-up process, in addition to the keys already required to manage data confidentiality and authenticity. These additional keys are to be securely initialized, stored and updated, which has an impact on energy consumption.

In this paper, we propose a new way of generating unpredictable wake-up tokens, using a recursive calculation that depends only on previous tokens and messages already exchanged with the main radio. This approach does not require any of the energy-consuming features of the existing methods. Moreover, generating tokens recursively and with more data unknown to attackers ensures that our solution is more secure than existing ones. The main contributions of our work are summarized below.

- We present SWARD, a communication protocol for the wake-up radio which is compatible with many common communication protocols for the main radio.
- We describe a secure and energy-efficient method to generate wake-up tokens.
- We perform a security analysis which shows that our solution is resilient against both attackers gaining access to some secret data and brute-force attacks.
- We evaluate, in simulation, the energy and performance overhead of our solution. In particular, we analyse the effect of the length of wake-up tokens on the daily energy consumption of a node with and without Denial-of-Sleep attacks. Results indicate that with wake-up tokens of 32 bits, SWARD has an energy overhead of only 0.03% for a typical IoT node, while increasing its lifetime by more than 40

WiSec '18, June 18–20, 2018, Stockholm, Sweden

https://doi.org/10.1145/3212480.3212488

times under Denial-of-Sleep attacks when compared to a non-secure node.

The remainder of this paper is organized as follows. Section 2 briefly introduces the context and assumptions made in our work, then describes our solution to compute wake-up tokens and the proposed communication protocol with the wake-up radio. Then, Section 3 details the implementation choices and the security and performance evaluation of our solution. Finally, in Section 4 we present and analyse related work, and compare it to our solution.

2 SECURE WAKE-UP PROTOCOL

In this section, we describe first the assumptions made on the network and the nodes, then the method to compute recursively and initialize wake-up tokens, and finally the proposed process for wake-ups.

2.1 Preliminaries

We consider an IoT node comprising two subsystems, which we call "on-demand" and "always-responsive", as it can be seen on Figure 1. The on-demand part includes the main radio transceiver, a processor core, memory and potentially several coprocessors, and can be deactivated in order to save energy. The always-responsive part contains the ultra-low power wake-up radio, in charge of waking up the on-demand subsystem upon receipt of a wake-up token. As explained in [12], such an IoT node with two parts can be fabricated with commercially available products.

We assume that exchanges on the main radio are secured, which means they are both encrypted and authenticated, and their integrity is ensured. Therefore, potential attackers cannot know the content of the messages exchanged on this radio, nor can they alter them or forge malicious messages. This is not a strong assumption, as most of the standard communication protocols for the IoT such as Bluetooth Low Energy [14], Wi-Fi HaLow [10], LoRaWAN [1], or Low Rate Wireless Personal Area Networks based on IEEE 802.15.4 [13] (ZigBee, WirelessHART, ISA100.11a) support such secure communication. If communication on the main radio were not secure, attackers would be able to send requests continuously to it and prevent the node from going back to sleep mode; thus, mitigating Denial-of-Sleep on the wake-up radio would be useless. Symmetric encryption is used in all of these protocols to secure communication over the main radio. The mechanism to securely exchange the corresponding keys is handled by each communication protocol and is out of the scope of this paper.

Our protocol will be described for networks with a star topology, where only a central node or base station can wake up other nodes. Exchanges with the main radio are considered bidirectional: any node can both send and receive messages to and from the base station. We assume an acknowledgement is sent each time a node or base station receives a message, which is possible for all of the communication protocols cited previously.

2.2 Wake-up token generation

We assume that at the n-th wake-up, M messages are exchanged (sent or received) between two nodes, with $M \geq 1$. The k-th message exchanged during the n-th wake-up, with $1 \leq k \leq M$, is reffered to as $Message_{n,k}$. The wake-up token (WuT) for the next wake-up, WuT_{n+1}, is generated according to Equations (1) and (2):

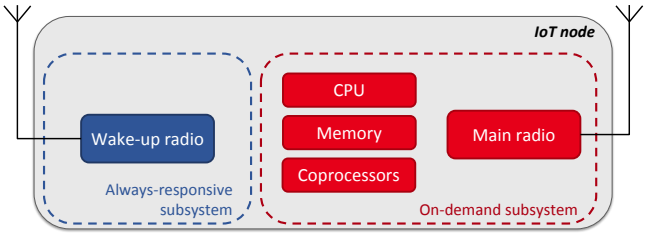


Figure 1: Representation of an IoT node separated into two subsystems.

$$WuT_{n+1,long} = hash(WuT_{n,long}||Message_{n,k}) \quad (1)$$

$$WuT_{n+1} = truncate(WuT_{n+1,long}) \quad (2)$$

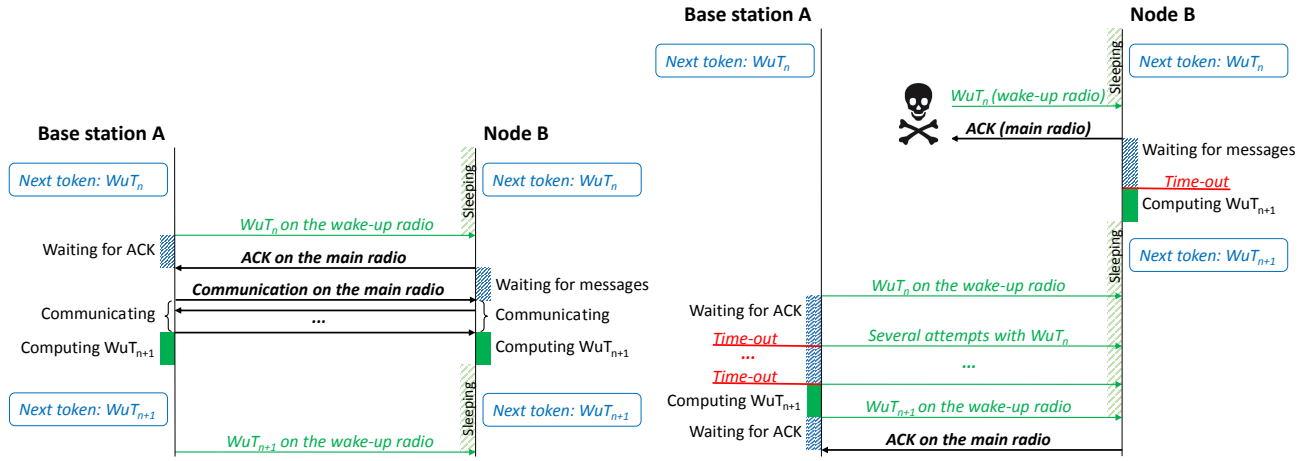
In Equation (1), *hash* is a cryptographic hash function, such as SHA-256 [5] or SPONGENT-88 [3], while || is the concatenation of the previous long version of the wake-up token with one or several of the messages exchanged during the n-th wake-up. For example, it can be the first ($k = 1$) or the last ($k = M$) message exchanged, or a concatenation of several messages; the choice of messages to consider for wake-up tokens computation, i.e. k, depends on the communication protocol on the main radio and has to be made for the whole network prior to any message exchange. Thanks to the use of a one-way hash function, it is computationally infeasible to obtain the original messages using the generated wake-up token. Therefore, an attacker intercepting this token is unable to retrieve any information about data exchanged between nodes.

In Equation (2), *truncate* means that the token that will be sent to the wake-up radio at the next wake-up, i.e. WuT_{n+1}, is the shortened version of the hash $WuT_{n+1,long}$. For example, WuT_{n+1} could be the first 32 bits of $WuT_{n+1,long}$. As *hash* is a cryptographic hash function, it is impossible, knowing only WuT_{n+1}, to guess the full $WuT_{n+1,long}$. An exhaustive study on the size of wake-up tokens and its effects on security and performance is presented in Section 3.

Though similar to a hash chain, the goal of this method is different, as it aims at preventing attackers from knowing both past and future values of this chain. Its security relies on a hash of two elements that are both unknown to attackers, $WuT_{n,long}$ and $Message_{n,k}$, as only short versions of wake-up tokens are sent to the wake-up radio while messages are encrypted before being exchanged with a secret key. A thorough security analysis is detailed in Section 3.1, and shows that this approach is resilient even with predictable messages.

2.3 Initialization

Equation (1) is recursive, as every wake-up token needs the previous one to be computed. Therefore, initialization has to be considered. As the first wake-up token must also be unknown to attackers, it has to be generated using secret information shared between nodes. As stated in Section 2.1, we assume communication is secure on the main radio, which means there is at least one secret key unknown to attackers. The first wake-up token is computed immediately



(a) Standard wake-up process

(b) Separate token update and resynchronization process

Figure 2: Message sequence chart between a base station A and a node B. Figure (2a) shows a wake-up process performed in standard conditions, while (2b) shows the token update and resynchronization process after an attacker was able to guess one wake-up token with brute-force attack.

after the network has been set up, which means that nodes are already able to communicate securely with each other, i.e. addresses and secret cryptographic keys have been generated. Using this cryptographic material, the first wake-up token in its long version $WuT_{1,long}$ is computed as follows:

$$WuT_{1,long} = hash(Secret\_key||Node\_address) \quad (3)$$

In this equation, *hash* is the same cryptographic hash function as in Equation (1), and *Secret_key* is a symmetric key used to secure communication on the main radio which is unknown to attackers by hypothesis. *Node_address* is the address of the node and ensures each node has a different wake-up token, even if several nodes share the same symmetric key. The first short wake-up token WuT_1 is then obtained with a truncation of $WuT_{1,long}$ according to Equation (2).

2.4 Wake-up process

Figure 2 presents the message sequence chart of communication and wake-up between a central base station A and a resource-constrained node B with our proposed approach. Initially, node B is in sleep mode, and the next wake-up will be the n-th. If A needs to communicate with B, it sends the wake-up token WuT_n, as shown in Figure 2a. When B receives it, it wakes up its main radio and sends an acknowledgement to A, which initiates the communication. According to the communication protocol used for the main radio, several messages can be exchanged downlink (from A to B) and/or uplink (from B to A). As soon as the whole communication is over, A and B each compute the next wake-up token using exchanged data. In order to calculate wake-up tokens, only messages that have received an acknowledgement can be considered by the sender node, while the receiver node can calculate the next token with any received message, after having sent the corresponding acknowledgement. On node B, the generation of the next token takes place in the on-demand part, which has more computing capabilities and a direct access to messages exchanged on the main radio. This token is then stored in the always-responsive part of node B, while the on-demand part goes into sleep mode, and the whole process can start again.

However, A and B may become desynchronized at some point, i.e. they will not have the same wake-up token anymore. This desynchronization may happen in two cases: (1) an acknowledgement is lost, including the wake-up acknowledgement, and (2) node B receives a wake-up token illegitimately, for example if an attacker found it by brute force or if two nodes in the network happen to have the same wake-up token. Note that as tokens are only calculated by the sender node upon receipt of an acknowledgement, lost messages cannot cause desynchronization: either messages are received by both nodes, or they are not considered to compute the next token. This ensures that even when nodes get desynchronized for any of the reasons cited previously, the messages used by one node to compute the next token are also known to the other node, and resynchronization will be possible.

As stated previously, if an acknowledgement is lost when nodes are communicating with the main radio, the corresponding message will not be used to calculate the next token by the sender node. Furthermore, if node B wakes up illegitimately, or if the wake-up acknowledgement is lost, there might be no communication between nodes, and B will go back to sleep after a time-out, which is described in Figure 2b. In both cases, it is likely that no valid message can be used to compute the next wake-up token. In that case, node B must still compute an unpredictable token and go back to sleep mode after a time-out. This next token, $WuT_{n+1,long}$, is computed with Equation 1 considering that messages are empty, which means that only the previous long token, $WuT_{n,long}$, is used as input to the hash function. This previous long token being unknown to attackers, they still cannot guess the next token, which will be explained more thoroughly in Section 3.1.

If desynchronization happens, A will temporarily try to wake up B with the wrong token. After several attempts without receiving

a wake-up acknowledgement, A will assume that its stored wake-up token is not the one used by B anymore, and calculate another possible token based on the underlying protocol with the main radio and on assumptions on the possible causes for desynchronization. A will send this token several times again, and repeat the process with other possible tokens, until it assumes that communication with B is broken. The calculation of possible tokens and the number of attempts with different tokens depend on the protocol used on the main radio; for example, the number of messages exchanged uplink and/or downlink and the robustness of the protocol regarding lost acknowledgements are important parameters to calculate possible tokens. All in all, the resynchronization process is entirely carried out by the base station A which is not as resource-constrained as the node B, while the latter is asleep and waiting for its next wake-up token. Therefore, the resynchronization process introduces no energy overhead for B.

3 IMPLEMENTATION AND EVALUATION OF THE PROPOSED SOLUTION

In order to evaluate the performance of SWARD, we implemented it on a circuit that we developed for energy-constrained IoT nodes in ASIC in technology 28nm FDSOI. The circuit includes both an always-responsive part with an integrated wake-up radio, and an on-demand part where power gating is used to switch to sleep mode. The main radio is external, though it can be deactivated as part of the on-demand subsystem. Our IoT node is powered with a 1000 mAh battery.

The always-responsive part includes the wake-up radio and a wake-up controller that initiates the wake-up of the on-demand part. Our wake-up radio has a variable data rate between 1 kbps and 1 Mbps and is able to cover all frequency bands from 433 MHz to 2.4 GHz with a sensitivity of -60 dBm, which makes it suitable to use with most RF protocols for the IoT. The average power consumption of the whole always-responsive part, including the wake-up radio, is 54.3 μW and has been obtained using post-synthesis simulation.

The on-demand subsystem consists of a main transceiver, a 32-bit RISC-V CPU [15] clocked at 150 MHz, sensors, and an AES coprocessor which is used to cipher and authenticate messages exchanged with the main transceiver. According to post-synthesis simulations, this part consumes on average 3 mW. As long wake-up tokens generated in the on-demand part are truncated before being stored in the always-responsive part and used for wake-ups, a lightweight hash function with a short hash size is sufficient to compute these tokens. Therefore, the hash function SPONGENT-88 which outputs a hash-code of 88 bits [3] has been implemented on the RISC-V. According to [2], SPONGENT requires about twice less memory and RAM than the more common hash algorithm SHA-256 [5].

In Section 3.1, we discuss the security of SWARD, and show that the complexity of attempting to guess the next wake-up token is prohibitive for attackers, using either knowledge of some secret data or a mere brute-force attack. Then, Section 3.2 provides an evaluation of the energy and time overheads of our solution through simulation for a typical use case for the IoT.

Table 1: Time needed to perform a brute-force attack on a single wake-up token with a data rate of 10 kbps on the wake-up radio, for several lengths of wake-up tokens

Wake-up token length	Time for a single brute-force attack
8	0.1 second
16	52.4 seconds
24	5.6 hours
32	2.6 months
64	$1.9 * 10^9$ years

3.1 Security analysis

As detailed in Equations (1) and (2), each wake-up token is generated using a hash of two elements that are both unknown to attackers: encrypted messages exchanged on the main radio, and the previous long wake-up token. Without these two elements, attackers are unable to calculate the next token.

First of all, we detail the security of our solution when all messages exchanged on the main radio since the first wake-up of the node are either empty or predictable, which is the best case for an attacker. Empty messages correspond to the situation when no messages are exchanged during a whole wake-up, as explained in Section 2.4. In that case, at the n-th wake-up, the only unknown quantity to attackers is the previous long wake-up token $WuT_{n,long}$. We assume attackers know all previous short tokens sent on the wake-up radio up to WuT_n. However, as SPONGENT-88 is a cryptographic hash function, it is computationally impossible, knowing only a truncation of $WuT_{n,long}$ such as WuT_n, to guess the full hash $WuT_{n,long}$. This is true as long as the size of $WuT_{n,long}$ is large enough compared to that of WuT_n: short tokens on 64 bits should be avoided when long tokens are 88-bit long, because the search space for a brute force attack on the remaining unknown bits would be reduced enough to make such an attack possible. Using a short token on 32 bits is secure enough, as a brute force attack on the remaining bits of the long token $WuT_{n,long}$ would still be much more expensive than the plain brute force attack on the short token WuT_n described below. All in all, with a 32-bit or less short token WuT_n, the only way attackers could predict the next token without performing an expensive brute force attack is to obtain the encryption key used to generate the first wake-up token $WuT_{1,long}$ and to calculate each token recursively.

The complexity of this attack further increases when messages exchanged on the main radio are unpredictable, and thus unknown to attackers without the adequate encryption key(s). In that case, attackers still need the original encryption key used to generate the first token $WuT_{1,long}$, but they also need all the messages exchanged on the main radio since the first wake-up of the node, which implies a continuous listening of the channel used by the main radio. In addition, attackers need the secret symmetric keys used to decrypt all of these messages, which is particularly challenging if more than one key have been used, for example if ephemeral session keys are used by the communication protocol on the main radio. Therefore, though such an attack is not strictly impossible, it is costly and complex enough to be prohibitive in most cases.

Most of the time, attackers have no knowledge of the secret key used by the AES to cipher messages exchanged on the main radio, and they have no way of guessing the previous long tokens. Therefore, they have to perform a brute-force attack on the current short token WuT_n to wake up the target node. The time needed to guess this token depends only on its length and the data rate of the wake-up radio. With a data rate of 10 kbps, the average time needed to brute-force a single wake-up token for various token sizes is shown in Table 1. On average, several months are needed to find a single 32-bit wake-up token, which makes the cost of brute-force attacks prohibitive. In addition, even if attackers managed to wake up a node by brute force without knowing its secret key, MAC-layer authentication on the main radio would reject any forged message and the node would compute a new wake-up token after a time-out, as described in Section 2.4, before entering into sleep mode again. Attackers would then have no choice but to start another brute-force attack to wake the node up again.

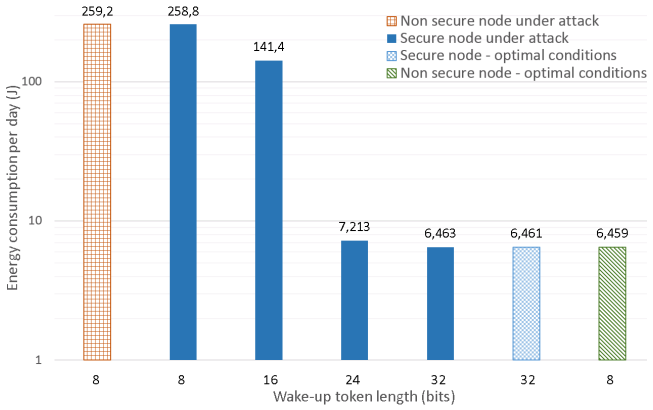


Figure 3: Energy consumption per day for our secure node with various token sizes, and for a non-secure implementation, with and without Denial-of-Sleep attacks.

3.2 Performance evaluation

Integrating SWARD into an IoT node introduces a small overhead on its energy consumption, as it requires to compute a new token in the on-demand part at every wake-up. In the following analysis, we assume that there are on average 10 genuine wake-ups per day for a node and that a node stays awake for 60 seconds, which is a reasonable use case for the IoT and the same than in related work by Liu et al. [11]. Moreover, we consider that the encrypted payload of the messages exchanged on the main radio is 16 bytes long, which is compatible, for example, with Bluetooth Low Energy [14] and IEEE802.15.4-based networks [13].

The time needed to compute a wake-up token with the hash function SPONGENT-88 on the RISC-V has been obtained with a cycle-accurate simulation at RTL level. In our target platform, 60.3 milliseconds are necessary to calculate each token. Therefore, the added energy consumption is 181 μJ. This overhead is independent of the wake-up token size, as all sizes are different truncations of the same hash-code. There is also a small energy and time overhead in the always-responsive subsystem: the non-secure node uses constant 8-bit wake-up tokens by default, while our secure node uses longer tokens that are updated at every wake-up. Nevertheless, this overhead is too small to be obtained with simulations.

With the node parameters and assumptions on the network described previously, Figure 3 shows the energy consumption per day for our secure node with various token sizes, as well as for a non-secure implementation with a constant 8-bit token, with and without attack. These values have been obtained with the assumption that no message or ACK is lost, which is an ideal situation; as the whole resynchronization process is performed by the base station, the energy cost for the node should be close to these numbers in real conditions. With a 32-bit token calculated at every wake-up, our solution decreases by more than 40 times the energy consumption of the node under Denial-of-Sleep attack compared to the non-secure implementation, while it increases its overall energy consumption by only 0.03% in case there is no attack. Under Denial-of-Sleep, the daily energy consumption of SWARD increases by only 0.06%. Therefore, 32-bit tokens are a good trade-off between the lifetime and performance of an IoT node both under Denial-of-Sleep attacks and in standard operating conditions.

4 RELATED WORK AND DISCUSSION

Several methods have been proposed until now to mitigate Denial-of-Sleep attacks. The first solution was presented by Falk and Hof [7]. In this approach, wake-up tokens are generated only on the node entering into sleep mode, and these tokens are then sent on a secure channel to other nodes in the network. This solution requires additional communication with the main radio in order to exchange tokens, which can be very energy-consuming. For more energy-efficiency, wake-up tokens should be computed locally on each node, using shared information and without any communication overhead, which is the approach selected in other related work as well as in our solution.

Another approach relies on one-time passwords (OTP), with two different implementations in [9] and [12]. OTP use time synchronization between nodes: each node regularly wakes up to compute a new wake-up token based on the current time, either with symmetric encryption or with a keyed-hash function. In [12], a new wake-up token is computed every 30 seconds by all nodes in the network, which is quite energy-consuming. Moreover, these mechanisms suffer from issues inherent to time synchronization in constrained devices, such as clock drift and resynchronization. Therefore, they might not be best suited for resource-constrained devices such as sensor networks.

Finally, another method to generate wake-up tokens as considered in [11] and [4] uses wake-up counters and secret symmetric keys dedicated to the wake-up process. The counter and the addresses of the sender and receiver nodes are either encrypted, or concatenated with a secret symmetric key and hashed. Using a secret symmetric key dedicated to the wake-up process is necessary, as the counter and addresses are easy-to-obtain data. According to simulations described in [4], this method is much more energy-efficient than the approach relying on one-time passwords.

Table 2 summarizes the main features of the methods proposed in related work and those of our solution. Methods relying on

Table 2: Summary of the main features of our solution compared to related work.

Feature	Secure exchange [7]	One-time password [9, 12]	Dedicated counter [4, 11]	**SWARD**
Additional communication	Yes	No	No	**No**
Management of extra symmetric keys	*Unknown*	Yes	Yes	**No**
Time synchronization	No	Yes	No	**No**
Lightweight algorithm	*Unknown*	[9]: Yes / [12]: No	No	**Yes**
Complexity of known-key attacks	Low	Low	Low	**High**

wake-up counters [4, 11] and one-time-passwords [9, 12] all require extra symmetric key(s) dedicated to the generation of wake-up tokens, in addition to the key that is already used to encrypt and authenticate data exchanged on the main radio. This implies the secure initialization, storage and update of additional symmetric keys, which is energy-consuming. If these additional symmetric keys are not managed securely enough, existing solutions would be vulnerable to Denial-of-Sleep attacks. SWARD, on the other side, does not require any additional key management, as it relies on the security of the encryption key used by the main radio to produce data unknown to attackers, i.e. the first long wake-up token $WuT_{1,long}$ and encrypted messages. As several independent elements unknown to attackers are required to calculate wake-up tokens, our solution is more secure than existing ones. Indeed, even if attackers were able to obtain the current encryption key used for the communication on the main radio, they would still need additional data to guess the next wake-up token, though other security issues would surely arise in that case. In Table 2, we refer to this added security of SWARD compared to related work as "complexity of known-key attacks".

In all previous related work, only [9] uses a lightweight algorithm, adapted for resource-constrained nodes, to compute tokens, while other implementations rely on traditional encryption and hash algorithms such as the AES [6] or SHA-256 [5]. Furthermore, existing solutions, especially those based on wake-up counters, may as well suffer from desynchronization issues as described in Section 2.4. SWARD proposes a more exhaustive wake-up protocol and a resynchronization process that solves these issues. All in all, our solution is more energy-efficient, more robust and more resilient to Denial-of-Sleep attacks than previous ones.

5 CONCLUSION

This article presents SWARD, a novel solution to prevent Denial-of-Sleep attacks on IoT networks that utilise wake-up radios. The solution has three main features: (1) it generates an unpredictable, new wake-up token at every wake-up of an IoT node, (2) the token computation is recursive requiring only local node information, namely the previous token and the messages exchanged securely on the main radio, and hence no extra information is exchanged between nodes, and (3) it integrates a protocol to keep nodes synchronized and to compute tokens when no messages are exchanged. Simulation results indicate that our solution has a negligible energy cost, while it greatly increases the lifetime of a node under Denial-of-Sleep attacks. Furthermore we carry out a security analysis that shows that our method is also slightly more secure than related work because it relies on several secret elements instead of only one. Our solution relies on the assumption that secure communication exists on the main radio, and hence it is dependent on the underlying communication protocol. We believe this is not a strong limitation, as most of the standard IoT protocols support secure communication, which is needed anyway to ensure confidentiality and authenticity and to prevent other types of Denial-of-Service attacks. Future work includes the integration of SWARD on an IoT node together with a standard communication protocol such as LoRaWAN, in order to evaluate more precisely its overhead under real Denial-of-Service attacks. The study of other hash functions to replace SPONGENT-88, and of hardware implementations of these functions, is another point for further investigation to enhance the energy-efficiency and/or security of SWARD .

REFERENCES

[1] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley. 2016. A Study of LoRa: Long Range and Low Power Networks for the Internet of Things. *Sensors* 16, 9 (2016).

[2] J. Balasch et al. 2012. Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices. In *Smart Card Research and Advanced Applications (Lecture Notes in Computer Science)*. Springer, 158–172.

[3] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede. 2011. spongent: A Lightweight Hash Function. In *Cryptographic Hardware and Embedded Systems - CHES 2011*. Springer, 312–325.

[4] A. T. Capossele, V. Cervo, C. Petrioli, and D. Spenza. 2016. Counteracting Denial-of-Sleep Attacks in Wake-Up-Radio-Based Sensing Systems. In *13th Annual IEEE Int. Conference on Sensing, Communication, and Networking (SECON)*. 1–9.

[5] Q. H. Dang. 2015. Secure Hash Standard. *Federal Inf. Process. Stds. (NIST FIPS) - 180-4* (Aug. 2015).

[6] M. J. Dworkin et al. 2001. Advanced Encryption Standard (AES). *Federal Inf. Process. Stds. (NIST FIPS) - 197* (Nov. 2001).

[7] R. Falk and H. J. Hof. 2009. Fighting Insomnia: A Secure Wake-Up Scheme for Wireless Sensor Networks. In *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. 191–196.

[8] V. Jelicic, M. Magno, D. Brunelli, V. Bilas, and L. Benini. 2012. Analytic Comparison of Wake-up Receivers for WSNs and Benefits over the Wake-on Radio Scheme. In *Proc. 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N '12)*. ACM, 99–106.

[9] A. Kavoukis and S. Aljareh. 2013. 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* 3, 1 (2013), 1–11.

[10] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin. 2015. A survey on IEEE 802.11ah: An enabling networking technology for smart cities. *Computer Communications* 58, Supplement C (March 2015), 53–69.

[11] J.-W. Liu, M. Al Ameen, and K.-S. Kwak. 2010. Secure Wake-Up Scheme for WBANs. *IEICE Transactions on Communications* E93.B, 4 (2010), 854–857.

[12] O. Stecklina, S. Kornemann, and M. Methfessel. 2014. A secure wake-up scheme for low power wireless sensor nodes. In *2014 International Conference on Collaboration Technologies and Systems (CTS)*. 279–286.

[13] J. Suhonen, M. Kohvakka, V. Kaseva, T. D. Hämäläinen, and M. Hännikäinen. 2010. Low-power Wireless Sensor Network Platforms. In *Handbook of Signal Processing Systems*. Springer, 123–160.

[14] R. Want, B. Schilit, and D. Laskowski. 2013. Bluetooth LE Finds Its Niche. *IEEE Pervasive Computing* 12, 4 (Oct. 2013), 12–16.

[15] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovíc. 2014. The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54* (May 2014).