HELIOS: Outsourcing of security operations in Green Wireless Sensor Networks

Giuseppe Ateniese*, Giuseppe Bianchi†, Angelo T. Capossele‡, Chiara Petrioli‡ and Dora Spenza‡
*Department of Computer Science, Stevens Institute of Technology, Email: gateniese@stevens.edu
†Department of Electrical Engineering, University of Rome Tor Vergata, Email: giuseppe.bianchi@uniroma2.it
‡Department of Computer Science, University of Rome "La Sapienza", Email: {capossele, petrioli, spenza}@di.uniroma1.it

***Abstract*—Energy-harvesting techniques for low-power embedded devices are opening up new opportunities for the design and optimization of security protocols for Green Wireless Sensor Networks. In this paper, we focus on scenarios where the energy resources of nodes in the network are heterogeneous, and propose a network-level solution that leverages the heterogeneity of harvesting capabilities to reduce the energy consumption of performing costly security operations. Our proposed distributed protocol, called HELIOS (Harvesting-EnabLed computatIon Outsourcing Scheme), allows nodes with scarce energy availability to outsource resource-demanding cryptographic operations to nodes that are harvesting power in excess, resulting in a significant reduction of their energy consumption.**

I. Introduction

The recent emergence of cost-effective low-scale power scavenging technologies is nowadays making possible to supplement the limited battery energy of wireless motes with energy gathered from their surrounding environment [1]. By providing virtually unlimited energy to nodes, energy harvesting techniques profoundly change the way Wireless Sensor Networks (WSNs) operate, and open up new possibility for the design and optimization of security protocols for low-power devices [2], [3], [4].

Precomputation-based security schemes are particularly well-suited to effectively exploit energy-recharging opportunities. In fact, when using energy scavenging technologies, nodes of a WSN experience significant changes in the power they harvest over time (e.g., due to varying weather conditions and seasonal patterns). The result is an alternation between periods in which energy must be sparely used, and situations in which there may even be an excess of energy available, which would be wasted unless used in the short term. Exploiting the occurrence of these occasional *energy peaks* is a promising approach for precomputation-based security schemes, as it permits to push part of the energy-costly security operations to excess energy periods [5], [6], [7].

In addition to support precomputations at a node-level, energy harvesting may also be exploited to enable practical optimizations on a network-wide scale [8]. Such optimizations are especially useful in hybrid networks of nodes with heterogeneous harvesting capabilities, composed by a combination of energy-scavenging nodes and traditional nodes powered by non-rechargeable batteries [9]. In addition, great variations in the energy reservoir of the nodes also occur in networks entirely composed by energy-harvesting motes, due to the fact that even physically co-located nodes may harvest energy at significantly different rates [10]. Such an energy imbalance in the network may result in excess energy being wasted by some nodes due to energy overflow, while others have little or no recharging opportunities.

To enhance system performance in these scenarios, in this paper we propose a distributed harvesting-aware scheme, called HELIOS (Harvesting-EnabLed computatIon Outsourcing Scheme), which outsources energy-costly security operations from nodes with scarce energy availability to nodes that are experiencing energy peaks. We provide a concrete application of HELIOS focusing on a practical security task: speeding up the generation of pairs of the form (r, g^r) (see Section II), which is generally the most expensive operation in Discrete Log based schemes [11]. In particular, we consider I-BPV, a scheme to efficiently generate Discrete Log pairs we recently proposed [12], which takes advantage of precomputation techniques and energy harvesting capabilities of Green WSNs. This paper provides the following contributions:

- We propose HELIOS, a distributed harvesting-aware scheme to outsource energy-costly security operations from nodes with scarce energy availability to nodes that are experiencing energy peaks;
- We discuss potential system vulnerabilities raised by the outsourcing of security-related operations, and present variants of HELIOS that allow to counteract the presence of malicious nodes in the network;
- We implement variants of HELIOS (including the small exponent test batch verification algorithm) on Telos B motes, and perform experimental measurements to accurately determine the time and energy consumption of security-related operations;
- We perform extensive simulations based on real-life measurements to show that delegating security operations in Energy-Harvesting WSNs can significantly improve the performance of the system, reducing the energy consumption of nodes with scarce energy resources of up to 98%.

The rest of the paper is organized as follows. Section II provides the background on the I-BPV generator. In Section III we present HELIOS and discuss two variants to counteract insider attackers. Performance of HELIOS are evaluated in Section IV. Section V concludes the paper.

II. Background: I-BPV generator

I-BPV is a scheme to reduce the cost of generating Discrete Log pairs by jointly exploiting precomputation and energy-harvesting capabilities embedded in modern sensor nodes [5], [12]. I-BPV has been employed in several security protocol implementations, e.g., to enhance the performance of a key agreement protocol based on ECMQV [13], to generate ECDSA (Elliptic Curve Digital Signature Algorithm) signatures in an energy-efficient way [5] and for the implementation of the Diffie-Hellman key exchange in DTLS protocols for the IoT [14]. In general, I-BPV can be used as a practical optimization by any protocol that requires the generation of pairs of the form (r, g^r) in a secure fashion.

I-BPV builds on the scheme proposed by Boyko, Peinado and Ventakesan in [11] (termed BPV in the following), and reduces its memory overhead to make it feasible on resource-constrained wireless sensor platforms.

The main idea of BPV is to precompute and store a set of n Discrete Log pairs, a subset of which is randomly chosen and suitably combined to perform costly modular operations with a significant computational gain[1].

Let $g \in \mathbb{G}_q$ be a generator of a cyclic group of order q. In what follows, unless otherwise specified, we resort to multiplicative notation. BPV speeds up the generation of pairs of the form (r, g^r), where r is a random number $\in \mathbb{Z}_q$, by preliminary precomputing (and storing in a table) a number n of randomly-chosen pairs. In particular, n integers $\kappa_1, \ldots, \kappa_n \in \mathbb{Z}_q$ are randomly generated. For each integer κ_i, the corresponding g^{κ_i} is computed, and the pair (κ_i, g^{κ_i}) is stored in a table T. Whenever a random pair (r, g^r) is needed, BPV and I-BPV randomly select a subset S of cardinality k out of the n precomputed pairs. The "random" value r is set as the sum of the k chosen terms, i.e.,

$$r = \sum_{i \in S} \kappa_i \mod q.$$

The corresponding g^r can then be computed by simply multiplying the corresponding precomputed values:

$$g^r = \prod_{i \in S} g^{\kappa_i} \mod q.$$

where g^{κ_i} is the precomputed value stored in T.

This algorithm is extremely efficient, as it performs a modular exponentiations by performing only $k-1$ multiplications. Indeed, multiplications are much cheaper to perform (in terms of both time and energy) than modular exponentiations. For example, on a Telos B mote, performing $k-1$ multiplications with $k = 16$ requires almost 1/10 of the time and energy required to perform a modular exponentiation [5].

The security of the BPV generator relies on the hardness of the *Hidden Subset Sum problem*, and it depends on its resistance to Birthday attacks [15]. To resist to Birthday attacks, in I-BPV the n precomputed pairs should be periodically *refreshed* (pairs refresh). In particular, a refresh operation should be performed after l runs of I-BPV, where l is a security parameter. In practical applications, the time interval with which the pool of pairs should be refreshed depends on both the frequency at which Discrete Log pairs are used (e.g., how often a ECDSA signature is generated) and on the desired level of security of the system.

However, performing pairs refresh is a energy-costly and time-consuming operation. For example, refreshing the whole pool of pairs when $n = 160$ costs more than 3 J of energy on a Telos B mote [5].

[1] We describe here the *simple BPV generator*. We refer the interested reader to [11], [5] for a detailed description of the *full BPV generator* and of I-BPV.

III. HELIOS

HELIOS (Harvesting-EnabLed computatIon Outsourcing Scheme) is a distributed harvesting-aware protocol that allows to outsource energy-costly security operations to nodes that are experiencing energy peaks. In this section, we describe how HELIOS can be used to outsource pairs refresh (which is the most energy-expensive operation required by I-BPV [5]) to nodes that are harvesting power in excess.

While this approach allows to reduce the energy imbalance in the network, delegating pairs refresh can lead to vulnerability of the system to insider attacks. In fact, if the delegated node is malicious or it has been compromised, it can generate "bad" pairs that can compromise the output of the delegating node. We thus discuss potential system vulnerabilities raised by the outsourcing of security-related operations, and present variants of HELIOS that allow to counteract the presence of malicious nodes in the network.

HELIOS provides a configurable solution to protect the system from this type of insider attacks. We start presenting HELIOS general operations in a trusted scenario, and then we detail two variants of HELIOS to counteract insider attackers. Table I summarizes the computational and communication overhead required by HELIOS in different classes of application scenarios, each associated to a given adversary model.

In all scenarios, we assume communication among nodes in the network is encrypted and authenticated. This guarantees protection against outsider attackers (e.g., eavesdroppers).

In the following, we also assume that nodes store the table T of precomputed pairs (Section II) in their flash memory.

A. Trusted scenario

In this scenario, we assume that insider attackers are not present, i.e., that every node in the network is trusted. Outsourcing is started by a node N_s that needs to refresh the n pairs (κ_i, g^{κ_i}) stored in its flash, only if its energy level is below a predefined threshold E_{low}. To delegate the energy-expensive modular exponentiations needed for the refresh, N_s dynamically selects a delegated node among neighbors that are experiencing an energy peak. Such nodes are excellent candidates for the outsourcing: they can perform modular exponentiations ideally at no cost, as the excess energy they

TABLE I
SUMMARY OF HELIOS CAPABILITIES AND OVERHEAD FOR DIFFERENT SCENARIOS.

Adversary Model	Protocol Capabilities	Computational Overhead	Communication Overhead
Trusted scenario	Outsourcing only	1	RTS + CTS + SEED + $n\times$ pairs
Up to $o-1$ malicious nodes colluding	Detection of attacks	$o \times n\times$pairs combination + verification	RTS + $o\times$(CTS + SEED + $n\times$ pairs)
Up to $o-1$ malicious nodes colluding	Detection of attacks and malicious nodes identification	$o \times n\times$ pairs combination + $o\times$ verification	RTS + $o\times$(CTS + SEED + $n\times$ pairs)

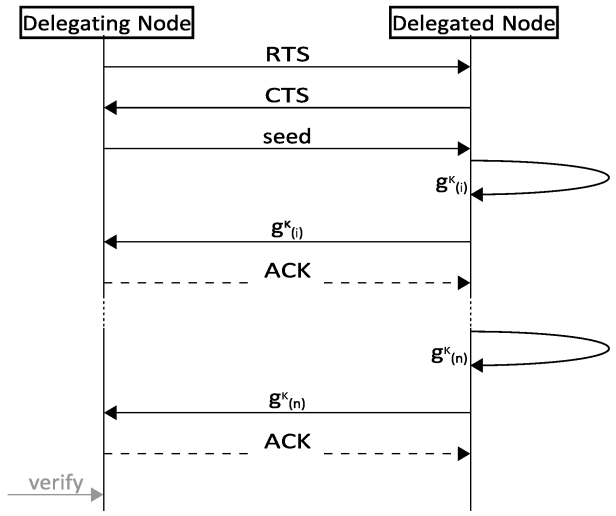


Fig. 1. Message flow diagram of HELIOS between delegating and delegated nodes.

gather during the energy peak would be wasted if not immediately used[2].

The delegated node is opportunistically selected in HELIOS through a timer-based contention. N_s starts the delegate selection by broadcasting a Request-to-Send (RTS) packet that contains its ID and the unique sequence number of the outsourcing request. Neighbors nodes receiving the RTS packet participate in the contention only if they are experiencing an energy peak and they are not serving another outsourcing request. Upon reception of an RTS packets, available neighbors respond with a Clear-to-Send (CTS) packet after a random jitter, which is computed based on their harvesting rate (i.e., the higher their harvesting power, the shorter the jitter). N_s selects as the delegated node N_d the first neighbor responding with a CTS to its RTS message. The selected node receives a message from N_s containing a seed, generated through a pseudo-random function, in our case an hash function H. This seed is used by N_d and N_s to generate the κ_i values: $\kappa_1 = H(seed||1), ..., \kappa_{n-1} = H(seed||n-1)$. The delegated node then starts generating (κ_i, g^{κ_i}) pairs. g^{κ_i} values are sent back to N_s and individually acknowledged. Note that there is no need to transmit the κ_i values over the network, as N_s can cheaply recompute them based on the seed. Once all the n pairs[3] are received, N_s resumes normal operations, using the newly obtained pairs for I-BPV (e.g., to sign messages).

Figure 1 summarizes the operation of the HELIOS protocol, and the messages that are passed back and forth between the delegating node N_s and the delegated node N_d to outsource pairs refresh.

In this version of HELIOS, named tHELIOS, the only energy cost incurred by N_s is that of communication with the delegated node. Such cost is at least an order of magnitude smaller than that of locally performing modular exponentiations (see Section IV-B).

However, depending on the considered scenario and adversary model, additional operations may become necessary. We discuss them in the following sections.

B. Non-trusted networks, malicious user detection

In this scenario, malicious insider nodes can be present in the network. This variant of HELIOS, called dHELIOS, allows to handle situations in which at most $o-1$ nodes collude with each others. In dHELIOS, the same operations described in Section III-A are performed, with the addition of a final verification step and with the generalization described in the following.

In a non-trusted environment, rather than selecting a single node as the delegate, N_s dynamically selects a *set* $\mathcal{D}_{N_s}$ of o neighboring nodes among those that are experiencing an energy peak. Selecting more than one delegated node is necessary in this scenario, because N_s should keep hidden the subset n in order to hold the *hidden subset sum problem* property (Section II). By choosing o nodes, N_s can obtain a hidden subset by combining the pairs generated by the delegated nodes with those generated by at least one non-colluding node.

As in tHELIOS, delegated nodes are opportunistically selected through a timer-based contention, in which the first o neighbors responding with a CTS to the RTS message sent by N_s are designated as $N_d^1, \dots N_d^o$. N_s then sends to each delegated nodes a different seed, which is used by each node in $\mathcal{D}_{N_s}$ to generate κ_i values. In addition to avoid transmitting κ_i values back to N_s, this step also ensures that potentially malicious delegated nodes cannot choose the κ_i values not randomly. Then, pairs generation and transmission starts. Once all pairs are successfully received, N_s combines the n pairs

[2]Depending on the application, other candidates for the outsourcing can also be nodes that are not experiencing an energy peak, but whose energy level is greater than a predefined threshold E_{high}

[3]For simplicity, we assume that all nodes in the network share the same value of n. If this is not the case, the RTS should include an additional field indicating the number of pairs that the delegated node must generate.

received from each of the delegated nodes to obtain the new pairs for its I-BPV generator, as follows:

$$(\kappa_i, g^{\kappa_i}) = \left(\sum_{j=1}^{o} \kappa_{ij}, \prod_{j=1}^{o} g^{\kappa_{ij}} \right) \mod q, \qquad \forall i = 1, \ldots n \tag{1}$$

As a last step, N_s must verify that the outsourced results are correct, i.e., it must check that each value a_d it received from delegated nodes is indeed $a_d = g^{\kappa_i}$. This step is needed to prevent that malicious nodes could generate incorrect pairs. To this end, N_s runs a probabilistic batch verification algorithm that allows to check a list of n pairs as a group (see Section IV-A). Such an algorithm gives a positive answer if all the modular exponentiations are correct, and a negative answer if at least one element of the group is invalid. The error probability of the batch verification is at most 2^{-v}, where v is a security parameter. With respect to the naive approach of performing n individual exponentiations, batch verification provides significant efficiency gains [16]. This last step allows N_s to detect whether there is malicious node in the set $\mathcal{D}_{N_s}$ of delegates. If it is the case, outsource results are discarded as they may be compromised. iHELIOS can then be run to identify the malicious nodes and blacklist them (Section III-C).

The energy cost sustained by N_s in dHELIOS is that of communicating with the delegated nodes and of batch-verifying the pool of combined results. With respect to tHELIOS, the communication overhead is increased by factor o. Computational overhead is also higher than that of tHELIOS, due to the need to verify the outsourced results.

C. Non-trusted networks, malicious user identification

Using dHELIOS allows to detect whether there is at least one malicious node in the set of delegates, but does not permit to identify which nodes in the set are misbehaving. Whenever an attack is detected, a new version of the protocol, called iHELIOS, can be employed to identify and blacklist malicious nodes. In order to do this, iHELIOS separately performs a batch verification of the pairs received from each of the o delegates, rather than verifying the combination of all the pairs, as done in dHELIOS. If the verification of the results received by delegate N_d^i returns a negative answer, N_s blacklists it, so that future outsourcing request started by N_d^i are ignored, and that N_d^i is not selected as a delegate anymore. Note that the blacklist maps every node's id with its respective public key (e.g., the same used to establish the secure and authenticated channel). As a consequence, a malicious node can not change its identity to bypass the blacklist. Only if all outsource results from each of the delegated nodes are successfully verified, N_s combines them to obtain the new pairs for its I-BPV generator (Equation 1).

iHELIOS is the version of the protocol with the highest overhead. In fact, to identify malicious nodes, iHELIOS additionally performs o individual verifications of the outsourced results with respect to dHELIOS. In a realistic setting, iHELIOS is expected to be used only upon need, to identify and blacklist malicious nodes when an attack is detected. It is worth noting that, once a node is blacklisted, all CTS messages received from it are discarded. This prevents a possible attack in which malicious nodes can outcompete honest sensor nodes by always choosing a zero-delay jitter to send a CTS, thus forcing N_s to always offload precomputations to them.

TABLE II
TIME AND ENERGY COST OF A SMALL EXPONENT TEST ON A TELOS B MOTE FOR $n = 160$ WHEN VARYING THE SECURITY PARAMETER v.

v (bits)	Time (s)	Energy (mJ)
4	28.5	154.15
8	51.4	277.70
16	80.2	433.22
32	137.3	741.67
64	247.9	1338.69

IV. PERFORMANCE EVALUATION

To evaluate the performance of HELIOS, we have measured both its computational overhead, expressed in terms of time needed to perform HELIOS operations, and its energy consumption on Telos B motes. Power consumption is derived via in-lab measurements. We experimentally evaluated the computational overhead by performing operations 10000 times on a Telos B mote, and recording the time needed to perform the overall cycle. This allows to estimate the average time needed to perform each operation.

A. Batch verification algorithm

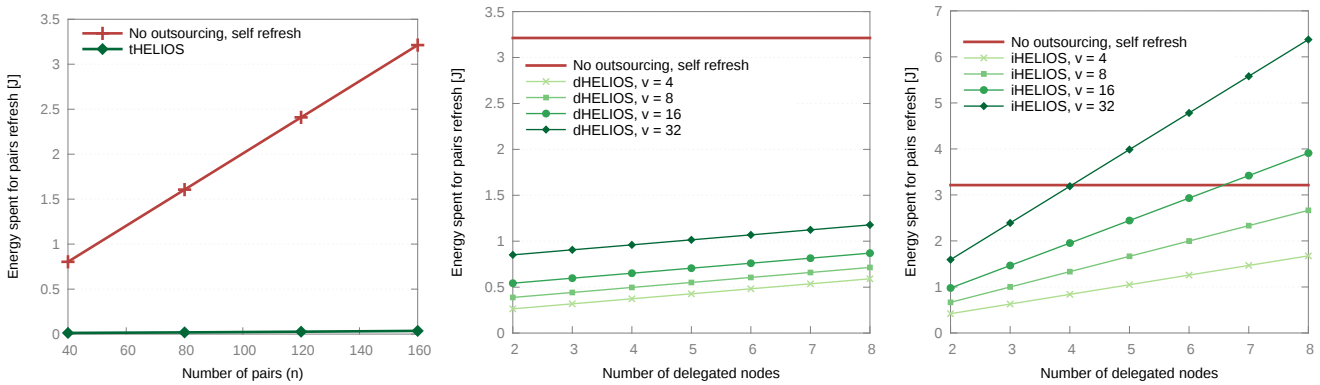
dHELIOS and iHELIOS make use of a probabilistic batch verification algorithm to check that the modular exponentiation results received by delegated nodes are correct (i.e., having received (g, κ, a_d) from a delegated node, check that $a_d = g^{\kappa}$).

More formally, let R a (polynomial time-computable, boolean) relation. The verification problem for R is: given an instance $inst$, check whether $R(inst) = 1$. In the batch verification problem a sequence $inst_1, ..., inst_n$ of instances is given and it is asked to verify that for all $i = 1, ..., n$ we have $R(inst_i) = 1$. A *batch verifier* is a probabilistic algorithm V which takes $inst_1, ..., inst_n$ and produces a bit as output. This output must be 1 when $R(inst_i)$ = 1 for all $i = 1, ..., n$. On the other hand, if there is even a single i for which $R(inst_i) = 0$, then $V(inst_1, ..., inst_n)$ should be equal to 1 with very low probability. Specifically, this probability should be at most 2^{-v}, where v is a security parameter.

Let g be a generator of a (cyclic) group G, and let q denote the order of G. The modular exponentiation function is $x \to g^x$, where $x \in \mathbb{Z}_q$. Define the exponentiation relation $EXP_{G,g}(x, y) = 1$ iff $g^x = y$, for $x \in \mathbb{Z}_q$ and $y \in G$.

A batch verifier is given a sequence $(x_1, y_1), ..., (x_n, y_n)$, for which we want to verify that $EXP_{G,g}(x_i, y_i) = 1$ for all $i = 1, ..., n$. The naive test is to compute g^{x_i} and test if it equals y_i, for all $i = 1, ..., n$. As this costs n exponentiations, the naive test is very expensive in terms of both time and energy.

To perform the verification in an energy-efficient way, we select the *small exponent test* [16] as the batch verification



(a) tHELIOS (b) dHELIOS (c) iHELIOS

Fig. 2. Comparison of the energy spent for pairs refresh with self-refresh and with different variants of HELIOS on a TelosB mote.

algorithm used in HELIOS. Our implementation picks bits $b_1, ..., b_n \in \{0, v\}$ at random. Then it performs:

$$x = \sum_{i=1}^{n} b_i \kappa_i \mod q, \quad y = \prod_{i=1}^{n} y_i^{b_i} \mod q,$$

and checks whether $g^x = y$. By keeping v relative small, a single exponentiation to a large exponent (e.g., g^x) is required, with a significant gain in terms of energy cost w.r.t. the naive test. With this setting, the error probability of the verification is 2^{-v}. The interested reader is referred to [16] for a detailed proof.

We implemented this batch verification algorithm on Telos B motes. Table II shows the cost of performing batch verification of a pool of 160 pairs by using the small exponent test. For comparison, employing the naive test to verify a pool of 160 pairs would require 592.16 seconds and 3.197 J.

B. Performance of HELIOS

In this section, we assess the performance of HELIOS by quantifying the energy spent by a TelosB mote to perform pairs refresh in different scenarios. Figure 2 shows the comparison of the performance of HELIOS with that of performing *self-refresh*, i.e., having the node to locally compute modular exponentiations to refresh the pool of the n precomputed pairs (Section II). Energy results include the energy spent for security-related operations (including operations for detection and identification for malicious nodes when required), as well as the energy spent for communication and to read/write from/to the node's flash. Figure 2(a) shows the energy spent by tHELIOS in a trusted scenario (Section III-A). The pool size n has been varied between 40 and 160. The energy spent by tHELIOS in such a scenario is always lower than $40\,\mathrm{mJ}$, which corresponds to more than 98% energy saving at the local node with respect to the self-refresh case in which outsourcing is not performed (which requires more than $3\,\mathrm{J}$ of energy). Figure 2(b) shows the performance of dHELIOS in a non-trusted scenario with malicious node detection. In this set of experiments, the size of the pool is set to $n = 160$. We varied the number of delegated nodes between 2 and 8, and the security parameter v of the small-exponent batch verifier between 4 and 32. The energy saving achieved by the local node when using dHELIOS with respect to self-refresh varies between 63% and 91%, depending on the number of delegate nodes and on the security parameter v. Figure 2(c) depicts the trade-offs of using iHELIOS in a non-trusted scenario with malicious node identification. As discussed in Section III-C, node identification allows to detect and blacklist malicious nodes. This provides higher level of security to the system, but also requires an higher energy cost. Results in Figure 2(c) allows to determine in which scenarios it is more convenient to use I-BPV with iHELIOS instead of using I-BPV with self-refresh. For example, when $v = 8$, using iHELIOS with up to 8 delegates allows to save energy with respect to performing self-refresh. However, if the security parameter v is set to higher values, using iHELIOS with the same number of delegate nodes does not provide any energy saving. Based on the security requirements of the considered scenario, i.e., parameter v and maximum number o of colluding nodes, the most energy-effective solution can be selected.

C. Performance of tHELIOS in different harvesting scenarios

For the last set of experiments, we run simulations using GreenCastalia [17], an open-source extension of the Castalia simulator [18] that we developed for accurate modeling of energy-harvesting WSNs. The energy model we use is that of TelosB, which uses the IEEE 802.15.4-compliant CC2420 transceiver. The default GreenCastalia settings were used for channel and radio models. 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. For accurate modeling of time and energy consumption of security-related operations, we extend GreenCastalia to include a realistic model of the microcontroller of TelosB motes, according to which time and energy consumption of security-related operations are modeled based on experimental measurements. In addition, we also provide simple models to account for time and energy spent to read/write from/to the flash and for sensing activity. We

accurately model the harvesting process by setting models parameters based on experimental measurements of prototype solar-powered nodes [19] and by using real-life traces of solar availability obtained from the National Renewable Energy Laboratory (NREL) at Oak Ridge, Tennessee [20]. A wake-up receiver (WuR) with nano ampere current consumption is used by nodes for energy-efficient on-demand communication. The wake-up receiver is modeled according to experimental measurements and specifications of a recent WuR prototype of ours [21]. 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.

In this set of simulations, we consider 20 solar-powered nodes that are randomly and uniformly deployed over a field of 100 x 100 m. We assume that insider attackers are not present, i.e., that every node in the network is trusted (Section III-A). Each node in the network performs temperature readings twice per minute, and sends them to the sink after signing them. When signing messages, I-BPV is used to generate ECDSA signatures in an energy-efficient way [5]. As discussed in Section II, the pool of pairs is periodically refreshed. Before performing a self-refresh, nodes with scarce energy availability run HELIOS to check whether they can offload pairs refresh to energy-rich nodes.

Different harvesting scenarios are simulated by varying the size of the solar cell powering the nodes, and thus the amount of energy they harvest. In particular, the power P_s harvested by a solar cell of size A and efficiency η is calculated as:

$$P_s = A \cdot \eta \cdot I,$$

where I is the radiant energy incident onto the cell surface. To simulate a realistic scenario, the cell efficiency is set to $\eta = 0.17$ [22], and we simulate shadowed areas in which the harvesting efficiency is reduced. Simulations are run for two weeks to account for varying weather conditions.

Figure 3 shows the energy spent for self-refreshes and the percentage of pair refreshes successfully delegated by a node placed in a shadowed area (thus with decreased harvesting efficiency and lower energy availability w.r.t. other nodes). As expected, having a bigger solar cell results in an higher occurrences of energy peaks, and thus in an increased number of delegated pairs refreshes. This results in a significant energy saving for nodes with scarce energy availability. In particular, when considering a solar cell of size 32cm^2, the energy spent for self-refreshes is approximately one third of that spent in a similar scenario in which nodes are equipped with solar cell of size 2cm^2.

V. Conclusions

We proposed HELIOS, a distributed protocol for precomputation-based schemes that allows to outsource energy-costly security operations from nodes with scarce energy availability to nodes that are harvesting power in excess. As security operations can lead to vulnerability of the system to insider attacks, we discussed variants of HELIOS to counteract the presence of malicious nodes in the network. Through simulations based on real-life measurements, we showed that outsourcing energy-costly security operations can significantly improve the performance of the system, reducing the energy consumption of up to 98%.

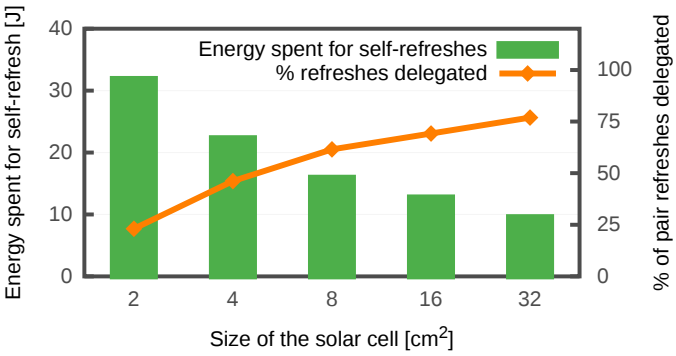


Fig. 3. Energy spent for self-refreshes and percentage of pair refreshes successfully delegated by a node using tHELIOS for different solar cell sizes.

Acknowledgments

This work has been partially supported by the MIUR project *SMARTOUR: Intelligent Platform for Tourism*, by the National Technological Cluster CTN001 0034 23154 *Social Museum and Smart Tourism*, funded by the Italian MIUR and by the European Union (European Social Fund) and by the *PON* project PON03PE00214. G. Ateniese gratefully acknowledges support from a Google Faculty Research Award and an IBM Faculty Award. D. Spenza gratefully acknowledges support from two Sapienza Research Grants for Young Investigators.

References

[1] S. Basagni, M. Y. Naderi, C. Petrioli, and D. Spenza, "Wireless sensor networks with energy harvesting," in *Mobile Ad Hoc Networking: The Cutting Edge Directions*. Hoboken, NJ: John Wiley and Sons, Inc., 2013, ch. 20, pp. 701–736.

[2] A. V. Taddeo, M. Mura, and A. Ferrante, "QoS and security in energy-harvesting wireless sensor networks," in *Proceedings of ICETE SECRYPT 2010*, Athens, Greece, July 26-28, 2010, pp. 1–10.

[3] V. Shakhov, S. Nam, and H. Choo, "Flooding attack in energy harvesting wireless sensor networks," in *Proceedings of ACM ICUIMC 2013*, New York, NY, USA, 2013, pp. 49:1–49:5.

[4] A. Di Mauro, X. Fafoutis, S. Mödersheim, and N. Dragoni, "Detecting and Preventing Beacon Replay Attacks in Receiver-Initiated MAC Protocols for Energy Efficient WSNs," in *NordSec*, 2013, pp. 1–16.

[5] G. Ateniese, G. Bianchi, A. Capossele, C. Petrioli, and D. Spenza, "Low-cost Standard Signatures for Energy-Harvesting Wireless Sensor Networks," *ACM Transactions on Embedded Computing Systems*, 2016.

[6] G. Bianchi, A. T. Capossele, C. Petrioli, and D. Spenza, "AGREE: exploiting energy harvesting to support data-centric access control in WSNs," *Elsevier Ad Hoc Networks*, vol. 11, no. 8, pp. 2625 – 2636, 2013.

[7] S. Pelissier, T. Prabhakar, H. Jamadagni, R. Venkatesha Prasad, and I. Niemegeers, "Providing security in energy harvesting sensor networks," in *Proceedings of IEEE CCNC 2011*, Las Vegas, Nevada, USA, January 9-12, 2011, pp. 452–456.

[8] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys Tutorials*, vol. 13, no. 3, pp. 443–461, Third 2011.

[9] S. Rao and N. Mehta, "Hybrid Energy Harvesting Wireless Systems: Performance Evaluation and Benchmarking," *IEEE Transactions on Wireless Communications*, vol. 13, no. 9, pp. 4782–4793, Sept 2014.

[10] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z.-L. Zhang, "Leakage-aware energy synchronization for wireless sensor networks," in *Proceedings of ACM MobiSys 2009*, New York, NY, USA, 2009, pp. 319–332.

[11] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up Discrete Log and Factoring Based Schemes via Precomputations," in *Proceedings of EUROCRYPT 1998*. Santa Barbara, California, USA: Springer, 1998, pp. 221–235.

[12] 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.

[13] A. Capossele, V. Cervo, C. Petrioli, and D. Spenza, "Counteracting Denial-of-Sleep Attacks in Wake-up-based Sensing Systems," in *Proceedings of IEEE SECON 2016*, London, UK, Jun 2016.

[14] A. T. Capossele, V. Cervo, G. De Cicco, and C. Petrioli, "Security as a CoAP resource: an optimized DTLS implementation for the IoT," in *Proceedings of the IEEE International Conference on Communications, ICC 2015*, June 2015.

[15] P. Nguyen and J. Stern, "The Hardness of the Hidden Subset Sum Problem and Its Cryptographic Implications," in *Proceedings of CRYPTO 1999*, vol. 1666, Santa Barbara, California, USA, 1999, pp. 786–786.

[16] M. Bellare, J. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Proceedings of EUROCRYPT 1998*, K. Nyberg, Ed., Helsinki, Finland, 1998, vol. 1403, pp. 236–250.

[17] D. Benedetti, C. Petrioli, and D. Spenza, "GreenCastalia: An energy-harvesting-enabled framework for the Castalia simulator," in *Proceedings of ACM ENSSys 2013*, Rome, Italy, November 14 2013, pp. 7:1–7:6.

[18] 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.

[19] A. Cammarano, C. Petrioli, and D. Spenza, "Online Energy Harvesting Prediction in Environmentally Powered Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 16, no. 17, pp. 6793–6804, Sept 2016.

[20] "NREL: Measurement and Instrumentation Data Center," 2011, http://www.nrel.gov/midc/.

[21] 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*, April 2015, pp. 522–530.

[22] IXYS Corporation, "XOB17 IXOLAR High Efficiency Solar Bits technical information," Jun 2009.