Low-cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Pre-computation Techniques?

G. Ateniese[+], G. Bianchi[*], A. Capossele[+], C. Petrioli[+]
+ Univ. Roma La Sapienza, ∗ Univ. Roma Tor Vergata
Rome, Italy
Email: ateniese, capossele, petrioli@di.uniroma1.it, giuseppe.bianchi@uniroma2.it

***Abstract*—Effective pre-computation techniques have been proposed almost 15 years ago for trimming the cost of modular exponentiations at the basis of several standard signature and key management schemes, such as the (Elliptic Curve) Digital Signature Algorithm or Diffie-Hellman key exchange. Despite their promises, the actual application of such techniques in the wireless sensor security arena has been apparently overlooked, and most of the research effort has rather focused on the identification of alternative *lightweight* constructions. However, modern sensor are equipped with relatively large flash memories which make memory consumption a less critical requirement, and emerging energy harvesting technologies provide occasional energy peaks which could be exploited for anticipating otherwise costly computational tasks. These trends push for a reconsideration of pre-computation techniques, which are explored in this paper as follows: (1) we further optimize prior pre-computation techniques by exploiting more recent results on Cayley graph expanders, (2) we implement an ECDSA scheme relying on pre-computations over two different wireless sensor node platforms (TelosB and MICA2), and (3) we experimentally assess the relevant performance and energy costs. In the traditional scenario of wireless sensor networks without energy harvesting, our prototype ECDSA implementation, despite still not fully optimized, outperforms prior work by almost 50%, and achieves an efficiency superior to NTRU signatures, natural candidates for low-power devices. Finally, (4) we quantitatively discuss ways to exploit harvested energy peaks to further improve efficiency.**

***Index Terms*—Sensor network security; secure communication architecture; digital signatures.**

I. Introduction

Wireless sensor networks (WSN) are nowadays being deployed in a large number of application domains [1], ranging from military environments and perimeter sensing [46], to weather and ambient control [18], industrial applications [24], power grids [19], health care [2], and so on. Security support appears of paramount importance in critical settings. Moreover, diverse scenarios call for different security requirements [47], [31], [42], [44] which are best supported by equipping the sensor nodes with a flexible set of security primitives rather than an one-size-fits-all security protocol. Indeed, some contexts may require secure point-to-point communication from sensor nodes to an infrastructure sink or between network node pairs; conversely, other deployments mandate for message integrity via digital signatures, e.g., when data source authentication is required.

The most crucial design aspects in wireless sensor nodes is *energy consumption*. The research community has been therefore challenged to envision cheap sensor node architectures, along with carefully crafted communication and sensing protocols, able to permit an extremely sparing usage of the available energy resources. Security support makes no exception to such a design strategy. Several among the current wireless sensor network deployments may operate unattended for extremely long periods, and may tackle contexts where even the physical access to a sensor, once released in the field, may be impossible. Clearly, the energy constraints in a WSN should *not* come along with weakened security protocols. For instance, if a critical application scenario mandates for the transmission of digitally signed gathered data to guarantee source authentication, we believe it is not appropriate to *weaken* such a requirement, e.g., by using HMAC message authentication with a key pre-shared across multiple sensors. Rather, security protocols and supported cryptographic primitives should be designed to retain effectiveness while using as little energy as possible.

One way to accomplish this design goal, a way we do *not* pursue in this work, consists of devising novel, *energy-friendly*, lightweight security primitives. The literature in the WSN arena is extremely rich in such a direction (see e.g., [29], [12], [48], [21], [32] to sample just very few recent papers, [33] for the first implementation of elliptic curve cryptography on WSN, and [13] for a comprehensive assessment of lightweight signature schemes considered appealing in WSN - a thorough state of the art in the area being well out of our scopes here). However, in security, a *novel* construction is not always advisable: despite its possible technical merits, acceptance of a novel approach requires time for a thorough scrutiny, and may require multiple revisions along this path (for instance, the NTRU signature was broken multiple times [17] when initially proposed). Moreover, real world sensor network deployers are more easily willing to leverage standardized security constructions, rather than novel approaches not challenged by a long-lasting real-world practice.

Precomputation techniques: In this paper we rather take a complementary way: how to *practically* achieve low cost security in real-world wireless sensor nodes, without requiring substantial changes in the security protocols set forth. This is

by itself far from being a new idea. Actually, a large amount of schemes based on pre-computation, and devised to accelerate standard digital signatures or key exchange protocols, were proposed as much as two decades ago [40], [6], [39], [27].

The scheme we mostly rely upon in this paper was proposed by Boyko, Peinado and Ventakesan in 1998 [5], and subsequently thoroughly investigated in [37], [38], as well as in [9], which extended it to the Elliptic Curve setting. We will refer to this scheme as BPV. The authors of [5] show the applicability of BPV to both RSA and Discrete log based schemes. Restricting, for convenience of presentation, to the latter case, the idea is to precompute and store a set of n Discrete Log pairs in the form $(k_i, g^{k_i} \mod q)$. A "random" pair $(r, g^r \mod q)$ is then generated by randomly choosing a subset of k terms k_i, setting $r = \sum k_i$, and computing the corresponding exponential term $g^r = \prod g^{k_i} \mod q$ via just $k-1$ multiplications, hence with a significant computational gain.

Despite its simplicity and appeal, to the best of our knowledge, neither BPV nor similar variants were so far considered in practical sensor node implementations. The reason seems apparently obvious and stems from the severe memory constraints of sensor nodes. As detailed in [38], the number of precomputed pairs must be sufficiently large to guarantee an acceptable level of security and thwart Lattice reduction attacks. Thus, BPV may require an amount of memory unavailable in a sensor node.

Technological trends: But is it true that modern sensor nodes are *severely* memory constrained? A 2009 comparative survey of wireless sensor node platforms [4] reveals that 4 out of the 8 analyzed platforms were already equipped with relatively *large* flash memories, ranging from 512 to 2048 KB. And such trend is deemed to increase, owing to the ever increasing application demands for local storage and processing of collected measurements, and the vanishing cost of flash memory chips. As a consequence, while a few tens of KB (in our specific case, we would need about 12 KB) to reserve for "just" an acceleration of security-related computations were simply unaffordable in first-generation motes, this extra memory consumption amounts to a relatively small fraction of the memory available over a today's mote.

Moreover, a second technological trend plays in favor of pre-computation techniques. Recently, cost-effective energy harvesting technologies have faced the WSN arena, in the form of techniques devised to supplement the battery power with energy gathered from the environment (e.g., solar, wind, etc. [45]). With harvesting technologies, energy peaks are occasionally available. In some cases the available energy can be even *excessive*, i.e., greater than the amount that can fit into the sensor node's supercapacitor, and thus it would be wasted if not immediately used. As a consequence, such emerging technologies very well fit pre-computation based schemes, as they permit to push part of the computation (or recomputation of already stored parameters for re-keying purposes) in the high/excess energy periods, and appear to be a very intriguing deployment playground for a variety of schemes leveraging an offline/online approach [15], [43], [23], [29].

Our contributions: In the paper, for concreteness, we specifically focus on ECDSA signatures (Elliptic Curve Digital Signature Algorithm), given their widespread consideration in emerging security protocols for low power devices (see, e.g., the IETF working groups ROLL and CORE-CoAP). In addition, we remark that NSA-approved products must employ ECDSA [1]. We focus on the signature generation cost given that verification is usually performed by infrastructure devices in WSN deployments. Our specific contributions are the following.

- We improve the memory overhead of the "full" BPV generator [5] using more recent theoretical results concerning Cayley graph expanders. As in the case of [5], the expander improves the randomness of the basic generator at the cost of extra storage, originally of the same magnitude of the precomputed table of n pairs. By applying such new results, we show that the extra storage can be reduced by a factor of 5.
- We implement ECDSA with pre-computations on two off-the-shelf sensor node platforms, TelosB and MICA2 motes, characterized by widely different design aspects, and provide an in-depth experimental assessment of the performance, energy cost, and emerging trade-offs. The resulting implementation performs a signature in 0.35 s and consumes about 8 mJ, outperforming the ECDSA implementations assessed in [13] on a comparable sensor node architecture (MICAz, using the same processor of MICA2), by almost 50%. Moreover, and perhaps surprisingly, our performance is superior to that of the $\text{NTRU}_{\text{SIGN}}$ scheme which is considered a quite natural candidate for low-power devices. This appears to be a very promising result, considering that our current implementation has still plenty of room for technical optimization.
- Finally, we experimentally quantify the amount of energy harvested through two technologies, micro solar cells and small wind turbines, which can be exploited for pre-computation purposes, and we preliminary show possible usages of harvesting capabilities.

II. Background

In this section we review known results which we further extended (see section III) and which we have exploited in our implementation and experimental assessment.

Pre-computation of Dlog pairs - simple generator: 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. Boyko, Peinado and Venkatesan first introduced in [5] a surprisingly simple technique for speeding up the generation of pairs of the form (r, g^r), frequently the most expensive operation in Discrete log based schemes. The scheme does not depend on the specifically chosen group, and, as shown in [9], it can thus be directly applied to

[1] http://www.nsa.gov/ia/programs/suiteb_cryptography/

the Elliptic Curve setting, and the relevant Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA) constructions.

The technique proposed in [5], hereafter referred to as *BPV generator*, speeds up the computation by preliminary precomputing, and storing in a table, a number n of randomly chosen pairs (k_i, g^{k_i}). Whenever a random pair (r, g^r) is needed, the generator randomly selects k out of the n precomputed pairs, sets the 'random' value r as the sum of the chosen terms k_i, and computes the corresponding term g^r, by simply multiplying the corresponding precomputed values g^{k_i}. More formally, the BPV generator is composed of two distinct phases illustrated in what follows.

Preprocessing:
Generate n integers $k_1, \ldots, k_n \in \mathbb{Z}_q$.
$\forall j \in (1, n)$, compute g^{k_j}.
Store the pairs (k_j, g^{k_j}) in a table.

Online Pair generation:
Randomly generate $S \subset [1, n]$ such that $|S| = k$.
Set $r = \sum_{i \in S} k_i \mod q$.
Set $g^r = \prod_{i \in S} g^{k_i}$ using the tabulated values g^{k_j}.
Return the pair $(r, g^r) = (\sum k_i, g^{\sum k_i})$.

The algorithm is extremely efficient, as it requires $k - 1$ multiplications only. Of course, the generated value r is not uniformly distributed. However, with an appropriate choice of parameters n, k, the distribution of the generated values is statistically close to the uniform random distribution. The reader may refer to [37] for a thorough quantification of such closeness.

Full generator with Random Walk: The above generator is further extended in [5] by combining it with a random walk on a Cayley graph expander. Hereafter, we refer to this extension with the name *full BPV generator*.

We recall that, intuitively, a graph is an expander if it is easy to reach any vertex from any other in very few steps. In other words, a graph is an expander when, starting from any initial probability distribution on its vertices, a random walk on the graph will rapidly converge to the uniform distribution on all vertices. Obviously, expanders are of practical interest whenever their degree is *low* but their expansion 'speed' is large. The expansion performance of a graph can be quantified via a (vertex) expansion parameter γ. Let S be an arbitrary subset of vertices, and let $N(S)$ be the set of neighbors of S, i.e., the set of vertices connected to elements in S through a graph edge. A graph V is a γ-vertex expander if, for any subset S of vertices, $|N(S)| \geq \gamma |S|(1 - |S|/|V|)$. Clearly, we wish to have $\gamma > 1$ as large as possible. Most of the results concerning expanders (including all results presented in the next section III) are expressed in terms of an alternative (spectral) parameter $\epsilon < 1$, an ϵ-spectral expander being a γ-vertex expander with $\gamma = 2/(1 + \epsilon^2)$.

The full BPV generator builds on a theorem proved by Alon and Roichman in [3]. Let $\mathbb{G}_q$ be a group of order q, and let $\mathbb{S}$ be a random set of group elements. Let $X(\mathbb{G}_q, \mathbb{S})$ be a Cayley graph of the group $\mathbb{G}_q$ with respect to a set $\mathbb{S}$ of elements. For any $1 > \epsilon > 0$ there exists a constant $c(\epsilon) > 0$ such that, for any random set $\mathbb{S}$ of $c(\epsilon) \log_2 q$ elements of $\mathbb{G}_q$, the Cayley graph is an ϵ-spectral expander almost surely.

Based on this result, the full BPV generator includes an *additional* table comprising n_e randomly chosen pairs. The two phases of the full BPV generator are modified as follows (n, k, g, q are defined as in the previous simple generator).

Preprocessing:
Generate n integers $k_1, \ldots, k_n \in \mathbb{Z}_q$.
$\forall j \in (1, n)$, compute g^{k_j}.
Store the pairs (k_j, g^{k_j}) in a table.
Further generate $n_e = c(\epsilon) \log_2 q$ integers $d_1, \ldots, d_{n_e} \in \mathbb{Z}_q$.
$\forall j \in (1, n_e)$, compute g^{d_j}.
Store the pairs (d_j, g^{d_j}) in a second table.
Initialize a value t to a random element in $\mathbb{Z}_q$.
Initialize a value T to g^{d_j}.

Online Pair generation:
Randomly generate $S \subset [1, n]$ such that $|S| = k$.
Select a random $d_i, i \in [1, n_e]$.
Set $t := t + d_i \mod q$ and $T := T \cdot g^{d_i}$.
Set $r = t + \sum_{i \in S} k_i \mod q$.
Set $g^r = T \cdot \prod_{i \in S} g^{k_i}$ using the tabulated values g^{k_j}.
Return the pair (r, g^r).

The generator has an extra cost in terms of storage due to the additional table (d_j, g^{d_j}) and the pair (t, T), and requires two extra multiplications in addition to the $k - 1$ ones. However, for an appropriate choice of $n_e \approx \log_2 q$, i.e., $c(\epsilon) = 1$, it permits to reduce the value k by a factor of two, i.e., the full generator with parameters n, k behaves as the simple generator with parameters $n, 2k$.

Application to ECDSA: The security of the generator relies on the hardness of the *Hidden Subset Sum problem*, studied in [38]: Given integers $M, b_1, \cdots, b_m \in \mathbb{Z}_M$, find $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}_M$ such that each b_i is some subset sum of $\alpha_1, \cdots, \alpha_n$ modulo M. This problem is conjectured to be hard if the ratio $n/\log_2 M$ is sufficiently large, more precisely greater than a given threshold approximately equal to 0.94.

As noted in [38], the reliance upon the Hidden Subset Sum problem holds also when the generator is used such that the integers b_i are not directly disclosed, but indirectly provided to a passive attacker via Discrete log terms such as g^{b_i}. This is indeed a case of significant practical interest, for instance when the generator is used for digital signatures such as ElGamal or the Digital Signature Algorithm (DSA). We devote the remainder of the section to discuss the specific application of the generator to Elliptic Curve DSA, which is of specific interest in this paper.

We recall that an ECDSA signature [22] is constructed as follows. Select an elliptic curve E defined over $\mathbb{Z}_p$ such that the number of points in $E(\mathbb{Z}_p)$ is divisible by a large prime q. Let $g \in E(\mathbb{Z}_p)$ be a point of order q. Let the integer $x \in [1, q-1]$ be a randomly chosen private key, and let the elliptic point $g^x \in E(\mathbb{Z}_p)$ be the corresponding public key (along with the public setup information q, E, g). Let $H(.)$ be a secure hash function.

ECDSA Signature for a message m:
select a random value $r \in [1, q-1]$;
compute the elliptic point $g^r = (x_1, y_1)$;
compute $w = x_1 \mod q$ (if $w = 0$, restart);
compute $r^{-1} \mod q$;
compute $s = r^{-1}(H(m) + xw) \mod q$ (if $s = 0$, restart).
The pair of integers (w, s) is the signature for message m.

Security of ECDSA relies on the choice of the integer r, which must be unique and unpredictable for each signature. Indeed, if r can be predicted, then it would be trivial to derive the secret key x from the linear modular equation

$$s = r^{-1}(H(m) + xw) \mod q \rightarrow$$
$$\rightarrow x = w^{-1}(sr - H(m)) \mod q$$

Similarly, if a same r is used for signing two different messages m and m', then the secret key x would be readily derived from the known signatures (w, s) and (w, s') by solving in the only unknown x the modular equation

$$(H(m') + xw)s'^{-1} = (H(m) + xw)s^{-1} \mod q \rightarrow$$
$$\rightarrow x = [H(m)s' - H(m')s] \cdot w^{-1} \cdot (s - s')^{-1} \mod q$$

When the truly random terms r are replaced with those produced by the generator, security of signature schemes such as Schnorr, ElGamal and DSS was shown in [38] to depend on a slightly modified variant of the Hidden Subset Sum problem, called the *Affine Hidden Subset Sum problem*, and which does *not* appears to be more complex than the original problem (the Lattice reduction attack in [38] being successfully adapted also to this problem): given a positive integer q, and $b_1, \cdots, b_m, c_1, \cdots, c_m \in \mathbb{Z}_M$, find integers $x, \alpha_1, \cdots, \alpha_n \in \mathbb{Z}_M$, such that each $b_i + xc_i$ is some subset sum modulo M of $\alpha_1, \cdots, \alpha_n$. Indeed, this obviously holds for ECDSA. It is sufficient to note that r, which, owing to the generator, it is a hidden subset sum, can be expressed as:

$$r = s^{-1}(H(m)+xw) = s^{-1}H(m)+xws^{-1} = b+xc \mod q$$

where, for each signed message, $b = s^{-1}H(m) \mod q$ and $c = ws^{-1} \mod q$ are known to a passive attacker.

III. Improved BVP generator

The BPV full generator combines the basic generator with a random walk on expanders based on Cayley graphs on abelian group. The distribution of the outputs of the basic generator is shown to be at most $2^{-(e+1)}$ statistically distinguishable from the uniform distribution, where $e = \frac{1}{2}\left(\log \binom{n}{k} - m\right)$ and $m = |p|$ for a prime p. Thus, for large values of $\binom{n}{k}$, the outputs of the basic generator follow essentially the uniform distribution. In BPV, the basic generator is improved by using expanders which will preserve randomness even when decreasing k. This is due to the Alon–Roichman theorem [3] which asserts that random Cayley graphs are expanders: for every $\epsilon > 0$ there exists a constant $c(\epsilon)$ such that the Cayley graph, obtained by selecting n_e elements independently and uniformly at random from a finite group G, has expected second largest eigenvalue less than ϵ (i.e., it is an expander with high probability), whenever $n_e \geq (c(\epsilon) + o(1)) \log |G|$. Because of this theorem, the value n_e is set to $c(\epsilon) \log |G|$ in BPV. Here the leading constant $c(\epsilon)$ is $4e/\epsilon^2$ which is about $10.87/\epsilon^2$.

The full BPV generator can be improved by showing that n_e can be smaller, thus saving in space. Our improved generator, which we call I-BPV, relies on the following results:

1) Landau-Russel [25] and Schulman-Loh [30] that showed that the $\log |G|$ term in n_e can be replaced by $\log D$, where $D \leq |G|$ is the sum of the dimensions of the irreducible representations of G. Note that a finite group G has only a finite number of irreducible representations (up to equivalence) and that $\sqrt{|G|} < D \leq |G|$.
2) Christofides-Markstrom [7] that showed that the constant $c(\epsilon)$ can be reduced from $10.87/\epsilon^2$ to $2/\epsilon^2$.
3) A proof that I-BPV is still safe against birthday attacks even though n_e is significantly smaller than previously intended.

We observe that, if G is a cyclic group of order q then there are exactly q inequivalent irreducible complex representations of G. Thus, in this specific case, $D = |G|$ but we can still benefit from the reduction of the constant $c(\epsilon)$. Namely, fixed an ϵ, the value n_e in I-BPV will be about $1/5$ of the n_e used in BPV. In practice, this means that the extra table stored in I-BPV will be five times smaller than the table in BPV, reducing the space overhead of the generator.

When the ratio $n/\log_2 q$ is in the order of 1 or more, based on [38], the security of the I-BPV generator depends on its resistance to birthday attacks, which directly derives from the relevant theorem in BPV.

Theorem [From [5]]: If G is a cyclic group of order q, then the expected number of repetitions in a run of I-BPV of length l is at most

$$\frac{\binom{l}{2}}{q} + \frac{l}{\binom{n}{k}}\left(\frac{1}{1-2^{-c}} + \frac{1}{c}k\log n\right)$$

for some constant c.

Our choice of parameters n and k is justified by the study of Phong Nguyen and Jacques Stern in [38], where they used the discrete Fourier transform to prove that the distribution of the BPV output is indistinguishable from the uniform distribution, and this holds without the addition of the expander.

IV. Experimental Platforms

To gain practical insights on the effectiveness of precomputation techniques, we have implemented, and experimentally assessed (in terms of performance and energy consumption), the ECDSA signature on two widely deployed off-the-shelf wireless sensor node families: TelosB and MICA2 motes. In the case of TelosB nodes, we could further carry out an integrated assessment of security mechanisms and energy harvesting technologies thanks to the availability of two testbeds,

Feature	M25P80	AT45DB
Erase	Slow (seconds)	Fast (ms)
Erase unit	Large (64kB-128kB)	Small (256B)
Writes	Slow (100 kB/s)	Slow (60 kB/s)
Write unit	1 bit	256B
Bit-errors	Low	Low
Read	Bus limited	Slow+Bus limited
Erase cycles	$10^4 - 10^5$	10^4
Energy/byte	1μJ	1μJ

Table I
MAIN FEATURES OF M25P80 (TELOSB) AND AT45DB (MICA2) FLASH MEMORY TECHNOLOGIES.

comprising nodes equipped with micro solar cells and wind microturbines, respectively. Technical details concerning the sensor node platforms and the harvesting technology employed are provided in the next subsections.

TelosB motes: TelosB [11] is a low power wireless sensor module developed and initially distributed to the research community by UC Berkeley and later on commercialized by Crossbow, inc. TelosB features an 8MHz MSP430 microcontroller, a 16b RISC processor, 10 kB of RAM, 48 kB of program memory (ROM), 1024 kB of external flash, and an IEEE 802.15.4 compliant wireless transceiver, the Chipcon CC2420.

The availability of relatively large storage flash memory chips embedded in modern sensor nodes plays a crucial role in permitting memory/performance trade-offs which were not possible in previous generation platforms due to the very scarce memory available. A flash memory is a specific type of EEPROM (Electrically Erasable Programmable Read-Only Memory) that enables access to n-bytes blocks in a single operation, instead of one operation per byte like conventional EEPROM memories. This memory is non-volatile, which means that energy is not needed to maintain the information stored in the chip. Telos B uses the ST M25P80 40MHz Serial code flash for external data and code storage. The flash memory holds 1024 kB of data and is decomposed into 16 segments, each 64 kB in size. The flash enables the random access for readings but shares SPI communication lines with the CC2420 transceiver. Flash erasure is block-oriented in that the minimum unit to be erased is a block. This means that all cells in a block must be erased together. Writing is performed on a per-byte basis, but it requires that the block be erased before writing on it.

MICA2 motes: The MICA2 Motes [10] are a second family of nodes commercialized by Crossbow, inc. MICA2 motes are equipped with a 4MHz Atmel ATmega128L 8b microcontroller, 4 kB of RAM, 128 kB of ROM, 512 kB of external flash and the Chipcon CC1000 low-power wireless transceiver.

MICA2 uses the Atmel AT45DB041 Serial DataFlash chip. Atmel AT45DB041 Serial DataFlash provides 512 kB of storage and it is divided into four sectors of 128 kB. Every sector is also divided into pages. Each page is 264 bytes long (256 bytes for data and 8 bytes for metadata). The pages can only be written or erased as a whole and, to maintain consistency, pages should be erased before being written. Unlike conventional flash memories, that enable random access to the data, this memory chip uses a serial interface and enables only sequential access. Table I compares the flash memory technologies used in the two considered platforms (M25P80 for TelosB, AT45DB for MICA2).

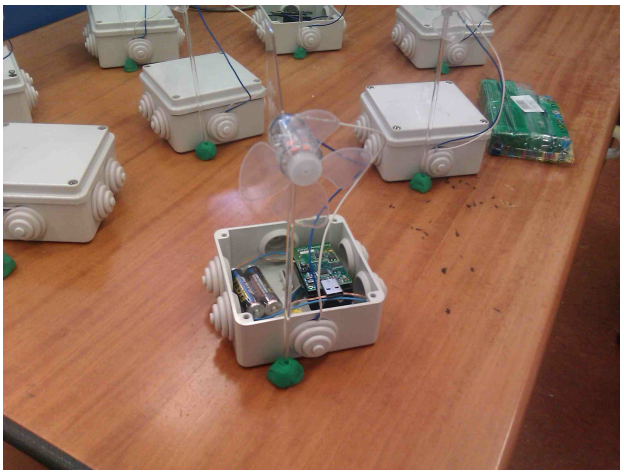


Figure 1. TelosB node with wind micro turbines.

Managing the flash memory: Care must be taken when accessing the flash in both read or write mode. In fact, in the considered sensor node platforms, the same SPI bus is used to access both the flash and the radio chip. We thus developed a software arbitration protocol to manage the SPI bus access in both of the two considered platforms. This access problem, of course, may not necessarily impact other platforms.

In order to manage the data stored in the flash, we relied on TinyOS 2.x primitives [26]. Specifically, TinyOS 2.x provides three basic storage abstractions: small objects, circular logs, and large objects. TinyOS 2.x divides the flash chip into one or more fixed-size volumes. Each volume provides a single type of storage abstraction (e.g., configuration, log, or block storage). The abstraction type defines the physical layout of data on flash memory. We used the LogStorage and ConfigStorage abstractions to write and read data.

Energy Harvesting Hardware: Experiments involving energy harvesting technologies were carried out using a testbed consisting of TelosB motes equipped with IXOLAR XOB17-04x3 micro solar cells [20] and a $10F$ Maxwell HC Power series capacitor [34]. We also integrated TelosB nodes with wind micro turbines as shown in Figure 1. In this case a $5F$ super capacitor from the same manufacturer is used. More details are discussed in Section V-C.

V. PERFORMANCE ASSESSMENT

In the next subsection we first provide some details on the optimizations employed in our ECDSA implementation. We then report, in section V-B, performance and energy consumption results for the criptographic primitives, considering both the case when they are run stand-alone and the case when the whole sensor is operational. Finally, section V-C

explores the usage of harvesting technologies and envisions some ways to further reduce the energy cost of the security related computations.

A. Implementation

We have implemented our solution in nesC for the operating system TinyOS 2.x [26]. In our implementation, based on the TinyECC library [28], the elliptic curve is an MNT curve, which can be written in the simplified Weierstrass form as

$$E(\mathbb{F}_p) : y^2 = x^3 + ax + b. \quad (1)$$

The elliptic curve E is defined over a prime field $\mathbb{F}_p$ where $p = 2^{160} - 2^{31} - 1$ as recommended by SECG [41]. According to NIST, this guarantees a security level of 80 bits.

Owing to the limited computational capabilities and the internal (RAM/ROM) memory constraints of sensor platforms, optimizations in the implementation are necessary. We used curve-specific optimizations to speed up modular multiplication and modular square, applicable to our case of group size p being a pseudo Mersenne prime.

To decrease the high computational cost to perform a modular inversion, we implement elliptic curve operations in projective coordinates using Jacobian representation. The affine coordinates can be transformed into projective coordinates which use three elements to represent a point (X, Y, Z), allowing the numerator and the denominator to be calculated separately. The elliptic curve defined in (1) is converted to Jacobian coordinates as follows:

$$E(\mathbb{F}_p) : Y^2 = X^3 + aXZ^4 + bZ^6, \quad (2)$$

where $X = xZ^2$, $Y = yZ^3$.

We used the OS function to generate randomness. We also experimented with PRNGs and both HMAC-SHA1, as a PRF, and SHA-512 truncated at 384 bits to behave like a "random oracle".

B. Performance

To assess the practical feasibility of our solution we have computed its computational overhead, expressed in terms of time needed to perform the needed operations. We have also evaluated the energy consumption associated with the operations performed by our solution.

Methodology: Energy consumption can be evaluated by means of the formula $E = U * I * t$, where t is the time to perform an operation, U is the voltage and I is the current intensity. The time t has been experimentally evaluated by performing tests which execute the selected operations 10.000 times, and recording the time needed to perform the overall cycle. This allows us to estimate the average time needed to perform each operation. The values U and I may be derived from both datasheet values as well as actual measurements. We have considered both: measurements were performed by taking the voltage difference between the sensor and a set of resistors used to clean up the signal from the noise coming from the lab environment. During the measurements the sensor was powered by a 3V generator. We have observed a negligible difference between the actual measurements and the values reported in the data sheet for both the platforms. For MICA2 motes, when the processor is in active mode, I = 8mA. TelosB instead consumes I = 1.8mA when in active mode. Typically, U = 3.0V with two new AA batteries.

	Telos B	MICA2
Exponentiation	3701ms/19.98mJ	2244ms/53.85mJ
Multiplication	193ms/1.04mJ	130ms/3.12mJ
A/P/A conv.	179ms/.	121ms/.

Table II
COMPUTATIONAL OVERHEAD AND ENERGY CONSUMPTION OF ECC OPERATIONS ON TELOS B AND MICA2 MOTES.

Cost of atomic Elliptic Curve operations: Table II shows the computational overhead and energy cost of ECC operations over the TelosB and MICA2 platforms, respectively. Consistently with the previous sections, we use multiplicative group notation rather than the more established additive one. The two basic operations are exponentiation (i.e., computation of an EC group point g^s with s a randomly chosen integer in $[1, q-1]$) and multiplication between two randomly chosen group points. Exponentiation is used in ordinary ECDSA, whereas our scheme only uses multiplications.

Exponentiation is, as expected, the most expensive operation. Table II shows that one exponentiation is executed in about 3.7s over a Telos B mote, and in about 2.2s over a MICA2. The difference between these values is due to hardware differences between the two platforms, as well the internal distinct optimizations of the assembly code. The energy consumption associated with exponentiation is 19.98mJ and 55.mJ for TelosB and MICA2 motes, respectively. This large difference is mostly due to the different current intensity in the two platforms (1.8 mA for TelosB versus 8 mA for MICA2). Our experimental evaluation also shows that a multiplication roughly costs about a factor 20 less than an exponentiation, in terms of both time and energy. A multiplication requires 193ms with an energy consumption of 1.04mJ on TelosB motes, and 130ms with an energy consumption of 3.12mJ on MICA2 motes.

By themselves, these results might (erroneously) suggest that the saving in using precomputations might be limited to the case of up to about 20 terms. As shown later on in table III this is *not* the case, and, for instance, 60 multiplications are performed in almost 1/4 of an exponentiation time. Indeed, our implementation performs (faster) operations in the Jacobian projective coordinates. The cost in converting from affine coordinates to projective coordinates and vice versa (labeled as A/P/A conversion in table II) is thus a fixed overhead which applies once to both exponentiation and multiplication. This cost is non negligible: in terms of time, it accounts to 179ms for TelosB and 121ms for MICA2 sensor node platforms. Note that the conversion of coordinates affine $\rightarrow$ projective $\rightarrow$ affine accounts for almost all the cost of performing a multiplication, being the latter step (projective $\rightarrow$ affine) the dominant cost. Nevertheless, backward conversion to affine is recommended

as security may be affected by leaving results in projective coordinates (see Naccache et al. [36]).

Cost of the precomputation-based ECDSA: the cost in terms of memory, time, and energy consumption of an ECDSA signature relying on precomputation depends on the parameters used in the generator, namely the number n of precomputed pairs (k_j, g^{k_j}), the number n_e of the elements (d_j, g^{d_j}) comprising the set used for the random walk over the Cayley graph expander, and the number k of elements drawn at each signature.

The parameters n and n_e are only related to storage, and hence do not impact the cost of an ECDSA signature in terms of time and energy consumption. Rather, the parameter k is related to the number of multiplications to be performed and hence affects performance results.

Table III provides an overview of the various time/energy costs involved in an ECDSA signature based on precomputations, along with the cost of the whole signature, for four values of the security parameter k, with $n = 160$. Specifically the table reports, for each sensor node platform, the time and energy consumption needed to perform: i) the modular sum of the coefficients $k_j \in \mathbb{Z}_q$, ii) the product of the k corresponding elliptic points g^{k_j}, and iii) the total ECDSA signature cost. Results show that the cost, as expected, grows with the size of the parameter k, but it remains significantly lower than the cost of an exponentiation even for large k. Note that the first row for MICA2 is left blank because MICA2 does not support $k = 60$ entirely in RAM.

Comparison with other techniques: Table IV compares the performance attained by our ECDSA signature based on precomputations (parameters: $n = 160$, $n_e = 32$, and $k = 8$) with alternative signatures, as well as other ECDSA implementations. The data reported in the table for the schemes different from ours are adapted from [13], which provides a comparative assessment of the reported schemes when implemented over a MICAz sensor node. Note that the MICAz platform, used in [13], relies on the same micro-controller employed in the MICA2 platform. Thus, it is reasonable to directly compare the signature time of the other schemes running on MICAz motes to that of our scheme on the MICA2 mote. However, MICAz has a different energy consumption than MICA2. Hence, for a fair comparison, we adapted the data of [13] to the consumption parameters of the MICA2 platform. MICA2 and MICAz [10] are both platforms relying on the very same hardware and CPU, clocked at the same 7.4 MHz. Their (substantial) difference resides only in the wireless transceiver, whose consumption was (indeed for fairness) NOT accounted in the table. MICA2/z energy consumption equivalence for computation is easily verified by data sheets, and well established in relevant literature (for a recent example involving crypto computations, see table 2 in [16]). The remaining entries in the table are provided for the reader's convenience, and report the size —Sig—, in bytes, of the considered signatures, along with the size, in bytes, of the private key —k_{priv}— and the public key —K_{pub}—.

In the comparison, we further accounted for the fact that the n pre-computed pairs (k_j, g^{k_j}) and the n_e pairs (d_j, g^{d_j}) cannot be entirely stored in the RAM. Indeed, MICA2 motes have only a $4kB$ RAM, whereas each pair requires 63 bytes of memory: 19 bytes for the integers k_j, and 22 bytes for each of the two coordinates of the elliptic points g^{k_j}. Even if $2.8kB$ were in principle available (the implementation of our scheme requires $18.2kB$ of ROM and $1.2kB$ of RAM), we considered the worst-case approach of storing *all* the pairs in the flash memory. Access to the flash brings about an extra time/energy cost. Specifically, reading one pair takes 1.94 ms and causes an energy consumption of 0.023 mJ. This supplementary flash access overhead explains the slightly worse results with respect to the performance reported in Table III for the same setting of the parameters.

Table IV clearly shows that pre-computation permits to significantly increase the speed with respect to any other scheme reported in the table: our signature is *almost three times faster* than the best ECDSA implementation reported in the table, and *almost two times faster* than $\text{NTRU}_{\text{SIGN}}$. Similar improvements are shown also in terms of energy consumption, reduced by almost 50% when compared against the most energy-effective implementation alternative. Also, considering that our current implementation can be further improved, we believe that this result makes a significant case for (re)considering pre-computation techniques *for making standard signatures practical in the wireless sensor domain, rather than choosing alternative signature schemes*.

Finally, note that performance improvements would still remain significant even when using larger k values (expected performance may be estimated from the values reported in Table III and the cost of the flash memory access).

Integrated performance assessment: We have investigated, using the TelosB motes, the cost of signing messages out of the overall sensor node energy consumption, when considering operational nodes involved in sensing activities and transmission of collected data.

All the precomputed pairs, accounting to about 12 kB, are stored in the flash memory. This is a worst case scenario, as part of them could be eventually stored in the 10 kB RAM of the TelosB platform. The storage requirement is readily determined as follows. The number n of pairs (k, g^{k_j}), each using 63 bytes, must be set to a value not lower than 160, the size in bit of the Elliptic Curve group, to prevent lattice reduction attacks [38]. Thanks to our optimization, the number n_e of supplementary pairs for constructing the Cayley graph are set to 32, one fifth of the group size in bits. Hence, 192×63 bytes are used in total.

Energy consumption due to communication can be accurately approximated based on the actual duty cycle followed by the considered protocol stack. Current state of the art protocols such as [35], [14], [8] operate with good performance at

	Telos B			MICA2		
k	$\sum k_j \mod q$	$\prod g^{k_j}$	ECDSA sign	$\sum k_j \mod q$	$\prod g^{k_j}$	ECDSA sign
60	10ms/0.05mJ	1026ms/5.54mJ	1229ms/6.63mJ	/	/	/
30	5ms/0.03mJ	604ms/3.26mJ	802ms/4.33mJ	3ms/0.07mJ	381ms/9.14mJ	523ms/12.55mJ
15	2ms/0.01mJ	391ms/2.11mJ	586ms/3.16mJ	2ms/0.05mJ	252ms/6.05mJ	393ms/9.43mJ
8	1ms/ϵ	291ms/1.57mJ	485ms/2.62mJ	$\cong$1ms/$\cong$0.02mJ	191ms/4.58mJ	331ms/7.94mJ

Table III
ANATOMY OF THE ECDSA SIGNATURE COST - TELOSB MOTES

Author(s)	Scheme	ROM	RAM	—Sig—	—k_{priv}—	—k_{pub}—	t_{sign}	$E_{CPU}(t_{sign})$
Gura et al.,	RSA	7.4kB	1.1kB	128B	128B	131B	10.99s	263.8mJ
Liu et al.,	ECDSA	19.3kB	1.5kB	40B	21B	40B	2.001s	14.8mJ
Driessen et al.,	NTRUSign	11.3kB	542kB	127B	383B	127B	0.619s	22.3mJ
	ECDSA	43.2kB	3.2kB	40B	21B	40B	0.918s	22.0mJ
	XTR-DSA	24.3kB	1.6kB	40B	20B	176B	0.965s	23.2mJ
This work	ECDSA	18.2kB	1.2kB	40B	21B	40B	0.346s	8.1mJ

Table IV
COMPARISON WITH NTRUSIGN, OTHER OPTIMIZATIONS OF ECDSA, AND XTR-DSA - DATA TAKEN FROM [13], ENERGY COST ADAPTED TO THE MICA2 CONSUMPTION PARAMETERS FROM THE ORIGINAL ONES BASED ON A MICAZ MOTE

around 1% duty cycle[2]. For this reason, in our experiments we have considered that nodes follow a duty cycle $d = 0, 5\%, 1\%$. We have also considered two scenarios differing in the type of sensors integrated in the node : i) sensor node equipped with humidity and temperature sensors, and ii) videocamera taking shots at a rate of 60 frames per hour. In the first scenario the average energy consumption due to sensing operation is 1.65 mW, while in the second the sensing energy consumption is higher, equal to 44 mW. By varying the duty cycle and the type of sensors (and associated sensing cost) we can explore the parameter space of the different wireless sensor applications and understand better the energy cost incurred by our scheme in different settings.

Specifically, figures 2, 3, 4, 5 report the percentage of the per-day energy consumption associated with sensing and communication, signature generation, and pair (k_j, g^{k_j}) readings from the flash. Each figure displays three concentric circles which represent the impact of the different energy consumption components when the number of signatures a node performs per day varies between $300, 1000, 3000$. The 300 signatures case corresponds to the inner circle while the 3000 signatures per day case corresponds to the outer circle. Each set of figures displays results for a given pair (duty cycle, type of sensor and associated frequency of reporting). Three figures are displayed in each set, which correspond to different settings of the security parameter k.

Results displayed in Figures 2, 3 show that the relative cost of signatures is large, up to 40% with $k = 30$, for a high number of required signatures (3000) and a low duty cycle of 0.5%. The saving obtained by reducing k to 8 is appreciable, but the overall cost of signatures still accounts to about 25%.

[2]The duty cycle is defined as $d = \frac{T_{ON}}{(T_{ON}+T_{OFF})}$, where T_{ON} is the fraction of time when the transceiver is active, and T_{OFF} denotes the fraction of time when the transceiver is in a low power "asleep state". In the latter state the energy consumption is negligible but the node cannot transmit or receive messages.

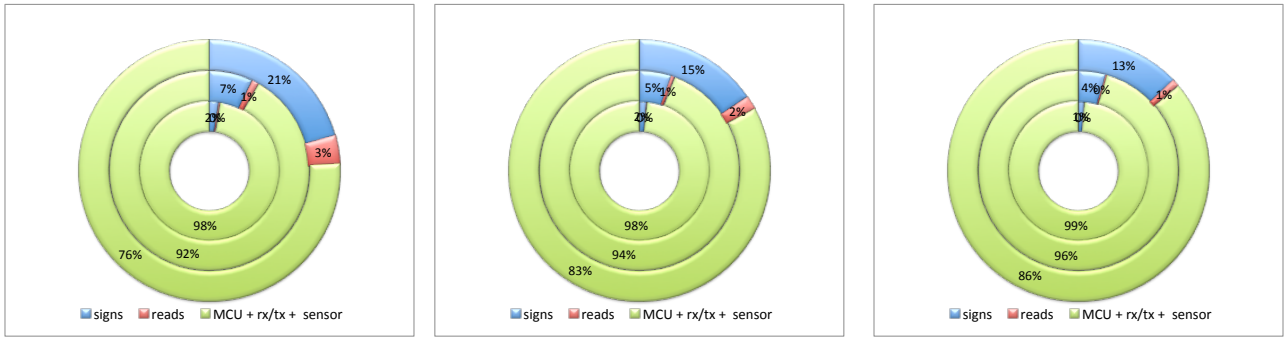
When the duty cycle is higher (see Figure 2) the cost of sensing and communicating increases, reducing the fraction of energy consumed by the signatures.

Figures 4, 5 show the case where the sensor node is equipped with a videocamera taking 60 shots per hour. In this case, the sensing cost is significantly higher, and the extra overhead required by the signatures is in the order of 10% or less even when considering a very large signature-generation rate.

C. Harvesting

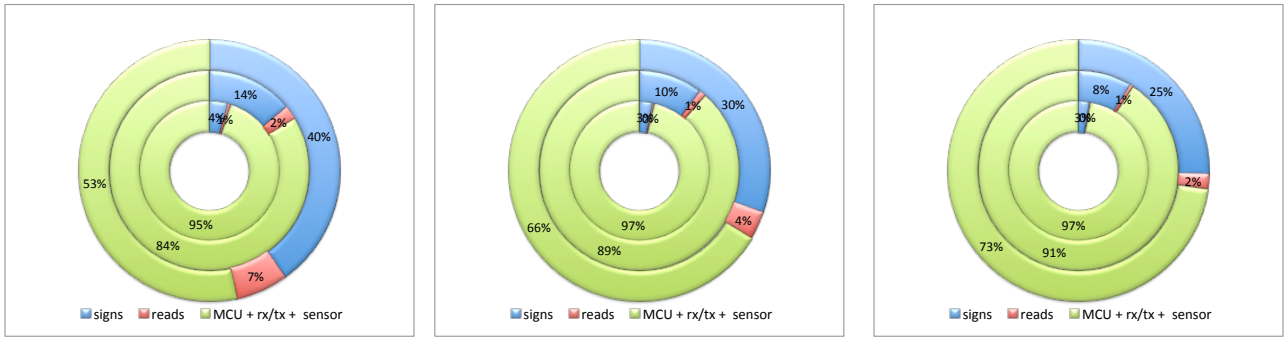
As a second set of experiments, we have investigated how energy harvesting can be exploited for precomputations. We have integrated TelosB nodes with XOB17-04x3 solar cells [20] and a $10F$ Maxwell HC Power series capacitor [34]. The motes were deployed close to the windows of a building for several months, at variable weather conditions. A dedicated TinyOS application was developed to periodically track the amount of energy generated by the solar cell. The resulting traces of harvested solar energy were used in our experiments (labeled as 'solar traces'). We also integrated TelosB nodes with wind micro turbines, as shown in Figure 1, and with a $5F$ super capacitor. Experiments were performed in this case for 3 months outside a building. Results run in the scenario of wind energy harvesting, based on real life traces, are labeled as 'wind traces'.

Since the supercapacitor suffers from leakage, energy which is harvested and not used progressively leaks and is wasted. Also, energy is lost if the supercapacitor is full and the harvested energy exceeds the node energy consumption. It is therefore convenient to precompute as much as possible values when harvested energy is available. To estimate how energy harvesting can be used to reduce the energy toll associated with security operations we run simulations using the real traces of harvested energy we experimentally obtained. Signatures are assumed to be uniformly distributed during the



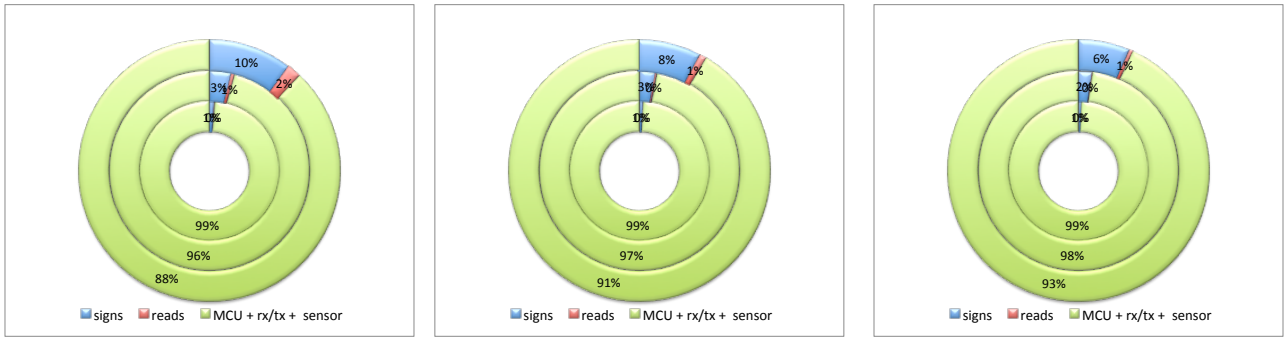
(a) $n = 160$ - $k = 30$. (b) $n = 160$ - $k = 15$. (c) $n = 160$ - $k = 8$.

Figure 2. Duty cycle 1% - Sensing temperature and humidity.



(a) $n = 160$ - $k = 30$. (b) $n = 160$ - $k = 15$. (c) $n = 160$ - $k = 8$.

Figure 3. Duty cycle 0.5% - Sensing temperature and humidity.



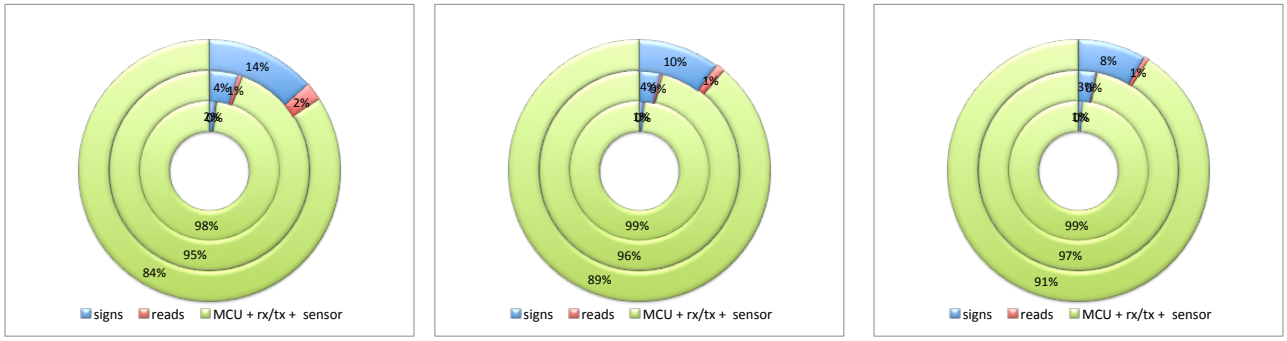
(a) $n = 160$ - $k = 30$. (b) $n = 160$ - $k = 15$. (c) $n = 160$ - $k = 8$.

Figure 4. Duty cycle 1% - Sensing camera.

day. Whenever there is harvested energy in the supercapacitor and there is free space in the RAM, we read (k_j, g^{k_j}) pairs from the flash, perform point addition and store the result in RAM, which can then be directly used to sign a future message. When we use a value stored in RAM to sign a message, the corresponding RAM memory is deallocated. We also exploit harvested energy, if available, to perform the actual message signature. Given the available RAM in TelosB nodes, we can precompute and store in RAM the values needed for up to $D_s = 81$ future signatures.

Table V displays the ratio between the energy drawn from battery to perform operations related to our scheme when energy harvesting is exploited, and the energy which would be needed to perform the same operations by a node with no harvesting capability. The scenario displayed in the table corresponds to the case when $n = 160$,$k = 8$. Results are shown for low power and high power sensing, and for wind and solar energy harvesting.

Results show that even a small solar cell such as the IXOLAR XOB17- 04x3, or a small wind turbine is able to



(a) $n = 160$ - $k = 30$. (b) $n = 160$ - $k = 15$. (c) $n = 160$ - $k = 8$.

Figure 5. Duty cycle 0.5% - Sensing camera.

	solar		wind	
signs	temp. & hum.	video	temp. & hum.	video
300	0.06	0.44	0.74	0.91
1000	0.08	0.44	0.76	0.91
3000	0.12	0.47	0.80	0.92

Table V
RATIO BETWEEN ENERGY DRAWN FROM BATTERY WITH HARVESTING AND WITHOUT HARVESTING

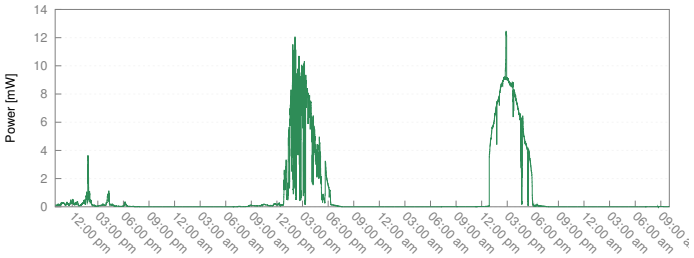


Figure 6. Solar energy harvested over three days.

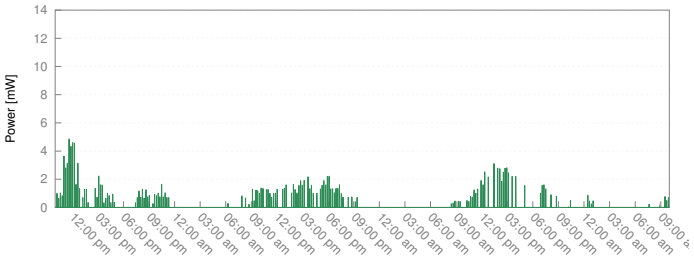


Figure 7. Wind energy harvested over three days.

provide enough harvested energy for the node to perform extensive pre-computations, reducing the energy consumption needed to sign messages. Reduction in the energy consumption ranges from 88% to 94% using temperature and humidity sensors, and from 53% to 56% when the node is equipped with a videocamera taking 60 shots an hour.

In the wind energy harvesting scenario the energy consumption reduction is more limited, ranging from 20% to 26% in case temperature and humidity sensors are used, and from 8% to 9% when using videocamera sensor.

The reason is in the lower amount of energy which can be harvested by exploiting wind microturbines. Figures 6 and 7 display the solar and wind energy harvested in three days, based on our traces. Wind generated energy is more constant over time, while solar energy is generated only during daytime but this effect is partially compensated by the availability of the supercapacitor for temporary energy storage. What makes the difference is therefore the overall amount of harvested energy over a typical day which is much higher when using solar cells than microturbines. This explains why we can achieve a higher reduction in battery energy consumption in the former case.

D. Harvesting and Full Exponentiations

Since we are using energy harvesting, one may ask: Why not computing directly a full exponentiation as part of pre-computation (with no increase in storage)?

The idea is that since we have extra energy, which would be wasted if not used, we could just compute pairs (k, g^k) and store them in RAM and FLASH until they are both filled up. When the pick of energy is over, we stop pre-computing pairs and start consuming them to generate signatures (one pair per signature, in this setting). We show next that this alternative solution would be much less performing. To do this, we must first fix certain parameters and devise a proper experiment.

Let's consider a node that is continuously signing messages, that is, the node keeps sensing and signing messages containing information captured by its sensors. The node may have only two states: (1) The first state is when the node has extra energy from harvesting. In this state, we assume the node's only task is to pre-compute and store pairs (k, g^k). This makes sense since the intention is to use energy in excess to populate the available memory with pairs. (2) The second state is when the node has no extra energy, namely when the supercapacitor is charging or discharging. In this state, the node will keep generating and transmitting signatures. In the event the node uses up all pairs, it will compute new ones on the fly. That is, the node first generates a pair (k, g^k) and then uses it to compute a single signature. The on-the-fly generation

	Naive			BPV		
	Precomputations	Signatures	FLASH	Precomputations	Signatures	FLASH
Day 1	6823	6823	0	19428	12726	6702
Day 2	77	77	0	0	597	6105
Day 3	3778	3778	0	6354	12459	0
Day 4	5302	5302	0	16038	13506	2532
Day 5	4758	4758	0	12936	15454	14
Day 6	5351	5351	0	17528	10783	6759
Day 7	5468	5468	0	15276	16664	5371
Average	4310	4310	0	11758	11532	2525

Table VI
COMPARISON OF NAIVE AND BPV-BASED APPROACHES

continues until the node reaches the first state again.

We run the two-state node above with pairs (k, g^k) computed using full exponentiation (naive approach) and our BPV-based technique (with $n = 160$ and $k = 8$). Results are shown in Table VI for a TelosB node with temperature and humidity sensing. In particular, the aim is to determine how many signatures the node can generate and transmit in a typical day. The days in the table are seven consecutive days taken from our traces while the average is computed over a month. In summary, with our approach the node can compute and transmit $11,532$ signatures per day as opposed to $4,310$ with the naive approach. Note also that with the naive approach, the node uses up all pre-computed pairs and does not make use of the FLASH. At the end of a typical day, the entire memory is completely empty. With our approach, instead, pair production is much faster than pair consumption, thus the node never needs to compute new pairs on the fly. In addition, at the end of a typical day, the node is left with a surplus of pairs (both in RAM and in FLASH) that can utilize afterward.

VI. CONCLUSIONS

In this paper, with focus on a concrete implementation of an ECDSA signature over two mote platforms (TelosB and MICA2) and its extensive assessment, we have shown that pre-computations permit to significantly reduce the energy cost and accelerate the speed of signatures in wireless sensor nodes. The extra memory cost, which we constrained to about 12 kB thanks to the application of new results on Cayley graph expanders, can be easily accommodated in flash memories which most of modern sensors currently employ.

We achieved an ECDSA-signature generation time below 350 ms over MICA2 motes, with an energy consumption below 10 mJ. We believe that, with further technical optimizations in the elliptic curve implementation, non-marginal additional improvements are possible. Our results have further shown that, with pre-computations, an ECDSA signature attains performance superior to lightweight approaches such as NTRUsign.

Finally, as a further argument in favor of pre-computation, we pointed out the emergence of energy harvesting technologies that opportunistically draw energy from the environment. In the paper, we provided an experimental quantification of the energy that micro solar cells and wind microturbines can make available to cryptographic processing. We believe that the exploitation of harvested energy for security protocols is a very compelling playground for future creative constructions.

ACKNOWLEDGMENT

This paper has been partially supported by the FP7 project GENESI (GrEen sensor NEtworks for Structural monItoring), by the ARTEMIS project #1000128 CHIRON (Cyclic and person-centric Health management: Integrated appRoach for hOme, mobile and clinical eNvironments) and by the PRIN project TENACE. G. Ateniese, in addition, gratefully acknowledges support from a Google Faculty Research Award and an IBM Faculty Award.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.

[2] H. Alemdar and C. Ersoy. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15):2688 – 2710, 2010.

[3] N. Alon and Y. Roichman. Random cayley graphs and expanders. *Random Structures Algorithms*, 5(2):271–284, 1994.

[4] R. Bischoff, J. Meyer, and G. Feltrin. *Wireless Sensor Network Platforms*. John Wiley & Sons, Ltd, 2009.

[5] V. Boyko, M. Peinado, and R. Venkatesan. Speeding up discrete log and factoring based schemes via precomputations. In *Advances in Cryptology - EUROCRYPT '98*, pages 221–235, 1998.

[6] E. Brickell, D. Gordon, K. McCurley, and D. Wilson. Fast exponentiation with precomputation. In *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Comp. Sci.*, pages 200–207, 1993.

[7] D. Christofides and K. Markstrom. Expansion properties of random cayley graphs and vertex transitive graphs via matrix martingales. *Random Structures Algorithms*, 32(1):271–284, 2008.

[8] U. M. Colesanti, S. Santini, and A. Vitaletti. *DISSense: An Adaptive Ultralow-power Communication Protocol for Wireless Sensor Networks*, pages 1–10. 2011.

[9] J. S. Coron, D. M. Raihi, and C. Tymen. Fast generation of pairs (k, [k]p) for koblitz elliptic curves. In *Selected Areas in Cryptography*, pages 151–164, 2001.

[10] Crossbow Technology. MICA2 mote platform datasheet. Document Part Number: 6020-0042-04.

[11] Crossbow Technology. TelosB mote platform datasheet. Document Part Number: 6020-0094-01 Rev B.

[12] G. Dini and I. M. Savino. Lark: A lightweight authenticated rekeying scheme for clustered wireless sensor networks. *ACM Trans. Embed. Comput. Syst.*, 10(4):41:1–41:35, 2011.

[13] B. Driessen, A. Poschmann, and C. Paar. Comparison of innovative signature algorithms for wsns. In *ACM WiSec*. 2008.

[14] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proc. 8th ACM Conf on Embedded Networked Sensor Systems*, SenSys '10, pages 1–14, 2010.

[15] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 263–275, 1989.

[16] D. Galindo, R. Roman, and J. Lopez. On the energy cost of authenticated key agreement in wireless sensor networks. *Wireless Communications and Mobile Computing*, 12:133–143, 2012.

[17] C. Gentry and M. Szydlo. Cryptanalysis of the revised ntru signature scheme. In *Proc. of the Int. Conf. on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, pages 299–320, 2002.

[18] A. Ghobakhlou, S. Shanmuganthan, and P. Sallis. Wireless sensor networks for climate data management systems. In *18th World IMACS / MODSIM Congress, Cairns, Australia 13-17*, 2009.

[19] V. C. Gungor, B. L. B. Lu, and G. P. Hancke. Opportunities and challenges of wireless sensor networks in smart grid. *IEEE Transactions on Industrial Electronics*, 57(10):3557–3564, 2010.

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

[21] R. Jiang, J. Luo, F. Tu, and J. Zhong. Lep: A lightweight key management scheme based on ebs and polynomial for wireless sensor networks. In *IEEE Int. Conf. on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–5, 2011.

[22] D. B. Johnson and A. J. Menezes. Elliptic curve dsa (ecsda): an enhanced dsa. In *Proc of the 7th conference on USENIX Security Symp.*, SSYM'98, 1998.

[23] M. Joye. An efficient on-line/off-line signature scheme without random oracles. In *Proc. of the 7th Int. Conf. on Cryptology and Network Security*, CANS '08, pages 98–107, 2008.

[24] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *3rd int. conf. on Embedded networked sensor systems*, SenSys '05, pages 64–75, 2005.

[25] Z. Landau and A. Russell. Random cayley graphs are expanders: a simple proof of the alon-roichman theorem. *Electronic Journal of Combinatorics*, 11(1), 2004.

[26] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In *in Ambient Intelligence*. Springer Verlag, 2004.

[27] C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In *Proc. of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 95–107, 1994.

[28] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 245–256, Washington, DC, USA, 2008. IEEE Computer Society.

[29] J. Liu, J. Baek, J. Zhou, Y. Yang, and J. Wong. Efficient online/offline identity-based signature for wireless sensor network. *International Journal of Information Security*, 9(4):287–296, 2010.

[30] P.-S. Loh and L. J. Schulman. Improved expansion of random cayley graphs. *Discrete Mathematics and Theoretical Computer Science*, 6:523–528, 2004.

[31] J. Lopez, R. Roman, and C. Alcaraz. Analysis of security threats, requirements, technologies and standards in wireless sensor networks. In *Foundations of Security Analysis and Design V*, volume 5705 of *Lecture Notes in Comp. Sci.*, pages 289–338, 2009.

[32] Y.-F. Lu, C.-F. Kuo, and A.-C. Pang. A half-key key management scheme for wireless sensor networks. In *Proc. of the 2011 ACM Symp. on Research in Applied Computation*, RACS '11, pages 255–260, 2011.

[33] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography, 2004.

[34] Maxwell Technologies. Datasheet HC power series ultracapacitors, August 2009.

[35] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proc of the 7th int. conf. on Information processing in sensor networks*, IPSN '08, pages 421–432, Washington, DC, USA, 2008. IEEE Computer Society.

[36] D. Naccache, N. Smart, and J. Stern. Projective coordinates leak. In *Advances in Cryptology - EuroCrypt 2004*, pages 257–267. Springer Verlag LNCS 3027, April 2004.

[37] P. Nguyen, I. Shparlinski, and J. Stern. Distribution of modular sums and the security of server aided exponentiation. In *Proc. of the Workshop on Comp. Number Theory and Crypt.*, pages 1–16, 1999.

[38] P. Nguyen and J. Stern. The hardness of the hidden subset sum problem and its cryptographic implications. In M. Wiener, editor, *Advances in Cryptology - CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 786–786. Springer Berlin / Heidelberg, 1999.

[39] P. Rooij. Efficient exponentiation using precomputation and vector addition chains. In *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 389–399. Springer Berlin Heidelberg, 1995.

[40] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[41] SECG. Sec 2: Recommended elliptic curve domain parameters version 2.0.

[42] J. Sen. A survey on wireless sensor network security. *CoRR*, abs/1011.1529, 2010.

[43] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Proc. of the 21st Int. Conf. on Advances in Cryptology*, CRYPTO '01, pages 355–367, 2001.

[44] S. Sharma, A. Sahu, A. Verma, and N. Shukla. Wireless sensor network security. In *Advances in Computer Science and Information Technology*, volume 86, pages 317–326, 2012.

[45] J. M. Weaver, K. L. Wood, and R. H. Crawford. Design of energy harvesting technology: Feasibility for low power wireless sensor networks. In *Proc. of the ASME 2010 Int. Design Engineering Tech. Conf. & Computers and Information in Engineering Conf. - IDETC/CIE*, 2010.

[46] M. Winkler, K.-D. Tuchs, K. Hughes, and G. Barclay. Theoretical and practical aspects of military wireless sensor networks. *Journal of Telecommunications and Information Technology*, pages 37 – 45, 2008.

[47] Y. Zhou, Y. Fang, and Y. Zhang. Securing wireless sensor networks: a survey. *IEEE Communications Surveys & Tutorials*, 10(3):6 – 28, 2008.

[48] T. A. Zia and A. Y. Zomaya. A lightweight security framework for wireless sensor networks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(3):53–73, 2011.