TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks

Leonardo B. Oliveira
UNICAMP, Brazil
leob@ic.unicamp.br

Michael Scott
Dublin City University
mike@computing.dcu.ie

Julio López
UNICAMP, Brazil
jlopez@ic.unicamp.br

Ricardo Dahab
UNICAMP, Brazil
rdahab@ic.unicamp.br

***Abstract*— Key distribution in Wireless Sensor Networks (WSNs) is challenging. Symmetric cryptosystems can perform it efficiently, but they often do not provide a perfect trade-off between resilience and storage. Further, even though conventional public key and elliptic curve cryptosystems are computationally feasible on sensor nodes, protocols based on them are not. They require exchange and storage of large keys and certificates, which is expensive. Using Pairing-based Cryptography (PBC) protocols, conversely, parties can agree on keys without any interaction. In this work, we (i) show how security in WSNs can be bootstrapped using an authenticated identity-based non-interactive protocol and (ii) present TinyPBC, to our knowledge, the most efficient implementation of PBC primitives for an 8-bit processor. TinyPBC is able to compute pairings in about 5.5s on an ATmega128L clocked at 7.3828-MHz (the MICA2 and MICAZ node microcontroller).**

I. Introduction

Wireless sensor networks (WSNs) [1] are ad hoc networks composed primarily of perhaps thousands of tiny sensor nodes with limited resources and one or more base stations (BSs). They are used for monitoring purposes, providing information about the *area of interest* to the rest of the system.

On the other hand, Pairing-Based Cryptography (PBC) [2], [3], [4], is an emerging technology that allows a wide range of applications. Pairings have been attracting the interest of the international cryptography community because it enables the design of original cryptographic schemes and makes well-known cryptographic protocols more efficient. Perhaps the main evidence of this is the realization of Identity-Based Encryption (IBE) [5], which in turn, has facilitated complete schemes for Identity-Based Cryptography (IBC) [6].

In the context of a WSNs, the issue of securing and authenticating communications is a difficult one, especially as currently nodes have no capacity for the secure storage of secret keys and are frequently deployed in unprotected areas, which make them more vulnerable to attacks [7]. One simple idea to introduce some kind of security is to fit each sensor node with the same cryptographic key to be used for all communications (e.g. [8]). But this does not authenticate the source of a message, and furthermore if one node is successfully attacked, all communications are compromised.

Assume now that there are n nodes, and that each has its own unique identifier (ID) from 0 to $n-1$. A better idea would be to fit each pair of nodes with a unique mutual key for all communications between them. But if that were the case each node would have to store $n-1$ secret keys, and furthermore $n(n-1)/2$ such keys would need to be generated in all. This is a big requirement in terms of time and storage for large n, especially considering that after deployment not all nodes will be in a position to talk with all other nodes. Furthermore, if new nodes are to be deployed at a later stage all existing ones must be recalled to be fitted with new keys.

Now consider this scenario: each node is issued with (i) a unique ID; and (ii) a unique secret, not shared with any other entity. Two parties, each knowing only the ID of the other and without communicating, are then able to derive a mutual secret unknown to any other party, and use that secret to derive a cryptographic key to secure their communications. It is also trivial to dynamically add new nodes to the the WSN without any impact on existing nodes

This scheme exists and in the world of Cryptography its known as the *Identity-Based Non-Interactive Key Distribution Scheme* (ID-NIKDS) [2]. It is *Identity-Based*, as only IDs are required – in particular no extra public key data is needed. It is *Non-Interactive*, as only the ID of the

"other" is required to determine the key – no interaction is required. And it is a *Key Distribution Scheme*, because each ends up with the same key value. Also, the protocol is authenticated as each party knows that only the other can possibly calculate the same key [1].

One issue has not been addressed – where does each entity get its unique secret from? It gets it from a *Trusted Authority*. This authority generates the unique secret from nodes IDs and a master secret of its own. Note that this "trusted authority" must be just that, as it is in a position to determine all the keys used within the system. It is our contention that such a set-up is an ideal way to bootstrap a WSN for security. The Trusted Authority is simply the *deployer* of the network, and there will be no issue in assuming their trustworthiness. Indeed it might even be regarded as a "feature" that the deployer is in a position to monitor all wireless traffic.

An alternative idea is to use the well-known Diffie-Hellman interactive key exchange to dynamically derive a mutual key between pairs of nodes. But this in not authenticated, and hence is subject to a deadly *man-in-the-middle* attack. Also interaction involves communication, and wireless communication is expensive in terms of power consumption.

Can the method we suggest be realized using regular public key cryptography? No it cannot. Indeed it is only relatively recently that a viable scheme has been discovered, and its implementation is quite difficult and computationally costly. However we only suggest it as a boot-strapping mechanism. Once the WSN nodes are deployed, they can cache keys, and create their own local keys for use within their own neighborhood. In this way the ID-NIKDS protocol need only be required very occasionally. Note that the ID-NIKDS secret is the only long-term secret that the node possesses, and that possession of such a secret is unavoidable if authentication is a requirement.

We do not claim that a scheme like this is by itself sufficient for securing WSNs. We do not claim that a network bootstrapped in this way will be immune from attack. An attacker could after all in theory compromise every node in the network. We do however claim that it is the best possible way to bootstrap a WSN, given that a node does not have secure storage for its secrets. Built on top of such a system, the network can dynamically evolve and develop routing and communications algorithms with maximum confidence that the damage caused by an attacker will be localized and minimized.

In this work, we firstly discuss why and how ID-NIKDS should be used to bootstrap security in WSNs. After that, we present TinyPBC, to our knowledge the most efficient implementation of PBC primitives for an 8-bit processor, and its performance on an ATmega128L (the MICA2 and MICAZ node microcontroller [9]). TinyPBC is based on Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL) [10] – which is a publicly available and open source library – and able to compute pairings, the most expensive operation of PBC, in about 5.5s. To sum up, our key contributions are:

1) demonstrate how sensor nodes can exchange keys in an authenticated and non-interactive way;
2) present the fastest pairing computation on an 8-bit platform; and
3) show the best figures for binary field multiplication on an 8-bit platform.

The remainder of this work is organized as follows. In Section II, we discuss the need for new security solutions in WSNs. We point out the synergy between IBC and WSNs in Section III. In Section IV we show how ID-NIKDS can bootstrap security in WSNs. Implementation and results are presented in Section V. Finally, we discuss related work and conclude in Sections VI and VII, respectively.

II. Bootstrapping Security in WSNs: Need for New Approaches

Security is mainly justified in WSNs because of their battlefield applications. We believe, however, that, once WSNs start to be deployed in large scale, security will become much more common than it is thought today. Apart from the well-known battlefield applications, confidentiality is likely to be a requirement in market and industrial scenarios. For example industries/farmers that employ WSNs to monitor their supply-chains/crops may want to keep their data private from competitors. Additionally, authentication might be useful even in domestic WSNs, avoiding interaction with nodes from a neighboring network.

Briefly, an ideal security scheme in WSNs should provide perfect connectivity and resilience. In other words, nodes should be able to (i) communicate securely with any other node they wish, and (ii) the compromise of a node should be restricted to itself. (Note that these properties should apply even to nodes deployed at different times.) Also the scheme should be low-cost in terms of both communication and computation.

[1]Actually, as we will see latter, an entity that is unconditionally trusted also can.

In WSNs, security is typically bootstrapped using key distribution schemes. Most of standard key distribution schemes in the security literature [11], however, are ill-suited to WSNs: conventional public key based distribution, because of its processing requirements; global keying, because of its security vulnerabilities; complete pairwise keying, because of its memory requirements; and those based on a key distribution center, because of its inefficiency. (See Carman *et al.* [12] for a good introduction to key distribution in WSNs.)

Symmetric key based distribution schemes (e.g., [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]) have been specifically designed for WSNs. While they are well-suited for the applications and organizations they were designed for, they might not be adequate for others. They provide a trade-off between connectivity and resilience, not providing an ideal level of both. Further, most schemes rely on some sort of interaction between nodes so that they can agree on keys.

More recently, it has been shown that alternative methods of Public Key Cryptography (PKC) are feasible in WSNs [25], [26], [27]. Because in those systems communicating parties only have a pair of keys, a private and a public key, PKC schemes are scalable and easy to use. This convenience, though, comes at a price: a way of authenticating public keys must be provided. And key authentication, in turn, whether traditional (PKI and/or certificates) or especially tailored to WSNs (e.g. [28]), often ends up in overhead – which is especially ill-suited to WSNs.

As we will show in Section IV, by using ID-NIKDS we are able to resolve these security issues.

III. Synergy between IBC and WSNs

PBC has paved the way for a new wide range of cryptographic protocols and applications [29]. It has also allowed many long-standing open problems to be solved elegantly. Perhaps the most impressive among those applications is IBE [5], which in turn has allowed complete IBC schemes [6], [2].

One may thus ask why IBC is still not widely deployed in security systems. Besides the usual time it takes for new technologies to be adopted in security systems, this is because IBC faces additional drawbacks. In particular they require a Trusted Authority (TA). A TA is an entity in charge of generating and escrowing users' private keys. That is, it is able to impersonate anybody else in the system. For that reason, a TA must be an entity that is unconditionally trusted by all network users. Such an entity, however, cannot be identified in many systems.

In WSNs, conversely, this is not a problem at all. The deployer – who loads software into nodes, deploys them in areas of interest, and observes collected data – is, obviously, trusted. In the world of WSNs, the deployer's role is represented by BS nodes. These nodes possess both laptop-level resources and physical protection. In other words, they can play the role of TA perfectly.

Another IBC's requirement is that the keys must be delivered over confidential and authenticated channels to users. If the cryptographic scheme is being used to bootstrap security – as very often is the case – there such channels will not exist. But again this is not a great concern to WSNs. In their security model, there is clearly a point in the time (i.e., prior to deployment) where secure channels between the BS and ordinary nodes do exist. Along with application software, nodes' private keys can be preloaded into nodes during the pre-deployment stage.

IV. Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks

The notion of Identity Based Cryptography dates back from Shamir's original work [6], but it has only become practical with the advent of PBC [4], [2], [5], [31]. The main idea is that known information that uniquely identifies users (e.g. IP or email address) can be used to derive public keys. As a result, keys are self-authenticated and additional means of public key authentication, e.g. certificates, are thus unnecessary. In this Section we define pairings (IV-A), and show how to setup IBC schemes in the WSN context (IV-B), and finally show how ID-NIKDS can be used so that nodes end up agreeing on keys. (IV-C).

A. Pairings: Definition

Bilinear pairings – or pairings for short – were first used in the context of cryptanalysis [4], but their pioneering use in cryptosystems is due the works of Sakai, Ohgishi, and Kasahara [2] *et al.* and Joux [31]. In what follows, let $E/\mathbb{F}_q$ be an elliptic curve over a finite field $\mathbb{F}_q$, $E(\mathbb{F}_q)$ be the group of points of this curve, and $\#E(\mathbb{F}_q)$ be the group order.

a) Bilinear pairing.: Let n be a positive integer. Let $\mathbb{G}$ be an additively-written group of order n with identity $\mathcal{O}$, and let $\mathbb{G}_T$ be a multiplicatively-written group of order n with identity 1.

[2]Today, however, other ways of providing IBE exist (e.g. [30]).

A *bilinear pairing* is a computable, non-degenerate function

$$e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$$

The most important property of pairings in cryptographic constructions is the bilinearity, namely:

$$\forall\, P, Q \in \mathbb{G},\ \text{and}\ \forall\, a, b \in \mathbb{Z}^*,\ \text{we have}$$

$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab}.$$

In practice, the group $\mathbb{G}$ is implemented using a group of points on certain elliptic curves and the group $\mathbb{G}_T$ is implemented using a multiplicative subgroup of a finite extension field. For more on pairing definitions, see for instance Galbraith [32].

Here we are using *Type 1* pairings (in the sense of Galbraith, Paterson and Smart [33]), and so we have the additional property that

$$e(P, Q) = e(Q, P).$$

In addition, pairings of *Type 1* permit strings to be hashed to a specific group. Those two aforementioned properties are required for efficient and simple implementation of the protocol (pairings of *Types 2* and *3* do not provide, at once, both properties).

B. Setup

To start up an IBC scheme, the TA first needs to generate and distribute private keys and public parameters. Broadly speaking, this procedure can be accomplished as follows in WSNs. Firstly, the BS generates a master secret key s and then calculates each node's private key. To do this it first maps each node's identity to a point on the elliptic curve, via a hashing-and-mapping function ϕ, so for node X, $P_X = \phi(id_X)$. It then calculates the node's private key as $S_X = [s]P_X$. It next preloads each node X with the following information: (i) the node's ID id_X, (ii) the node's private key S_X. Each node is also equipped with the function ϕ so that it can take any ID (e.g. id_Y) as input and outputs the public key corresponding to the ID (e.g. P_Y), [3].

Note that, besides the BS, only node X knows the key S_X.

[3]To be precise, a small number of non-secret public parameters are also needed to be stored into nodes, but for simplicity's sake, we will omit them.

C. Applying ID-NIKDS in WSNs

WSNs are composed of maybe thousands of tiny resource-constrained sensor nodes for which the scarcest resource is energy. Communication, on the other hand, is the activity that consumes most energy. This, in turn, means that besides meeting the needs described in Section II (i.e., perfect connectivity and resilience), an ideal key agreement scheme for WSNs should also not require any exchange of messages.

With the advent of PBC, however, a method of accomplishing this has come available. That is, PBC provides means to non-interactively distribute keys between any two network nodes, even if they were deployed at different times. Further, because nodes employ asymmetric primitives, the effect of node compromise is strictly local. In what follows, we show how the protocol due to Sakai, Ohgishi, and Kasahara, ID-NIKDS [2], can be employed to achieve such a goal. We assume that the setup protocol shown in Section IV-B has been already carried out.

Suppose two nodes A and B that know each other's IDs wish to decide on a secret key. Recall from Section IV-B that nodes A's and B's private keys are $S_A = [s]P_A$ and $S_B = [s]P_B$, respectively. Consequently, by bilinearity (Section IV-A) we have

$$\begin{aligned}\hat{e}(S_A, P_B) &= \hat{e}([s]P_A, P_B)\\ &= \hat{e}(P_A, P_B)^s\\ &= \hat{e}(P_A, [s]P_B)\\ &= \hat{e}(P_A, S_B)\\ &= \hat{e}(S_B, P_A).\end{aligned}$$

Note that A possesses S_A and can compute $P_B = \phi(id_B)$. Likewise, B possesses S_B and can compute $P_A = \phi(id_A)$. Therefore, both A and B are able to compute the secret key

$$k_{A,B} = \hat{e}(S_A, P_B) = \hat{e}(S_B, P_A).$$

(Formally speaking, a key derivation function must first be applied to $k_{A,B}$ in order to generate a key appropriate for cryptosystems. For details on this and other PBC protocols refer, e.g., to Paterson [29].) Additionally, A knows that only B – and the BS, a trusted authority – possess S_B and vice-versa, and consequently the protocol is authenticated.

Observe that due to the non-interactive nature of the communication nodes can agree on keys even if they are not online simultaneously. This is particularly useful

in WSNs, where nodes might follow sleeping patterns, may be deployed at different times, and often become temporarily unavailable due to physical obstacles or malfunctions.

Lastly, notice that we assume that nodes already know one another's IDs. And we understand that it is a reasonable assumption in WSN context. Since routing reports towards the BS already demands nodes to get to know their neighbor's IDs, exchange of IDs cannot be considered an overhead incurred by the key distribution protocol. Additionally, ID size is negligible when compared to public key and certificate sizes.

V. Evaluation

The utilization of pairings to implement security in WSNs is quite complex. For an 80-bit security level (RSA-1024 equivalent), as opposed to conventional Elliptic Curve Cryptography (ECC) – which works with 160-bit numbers – PBC works with 1024-bit numbers. In this section, we assess the costs incurred by PBC on an ATmega128L microcontroller, as included in the MICA2 and MICAZ node microcontroller [9]. The platform features an 8-bit/7.3828-MHz processor, 4KB of SRAM, and 128KB of flash memory (ROM).

A. Implementation

By far the most time consuming part when evaluating PBC protocols is the pairing computation itself [4]. In this section we present TinyPBC, an implementation of the η_T [35] pairing (pronounced "eta-t") for resource-constrained nodes – the source code is available at `www.lca.ic.unicamp.br/~loliveira#tinypbc`. Due to spaces constraints, we do not describe in depth all implementation details. Instead, we do point the reader out to more descriptive sources.

b) Security Requirements: To meet their needs for efficiency security requirements in WSNs are often relaxed. For example, some (e.g. [13]) have adopted a 64-bit security level. We, conversely, adopted a more conservative posture and thus used a 80-bit security level, as recommended by the NIST.

c) Pairing: The η_T [35] is possibly the fastest known pairing. It was proposed by Barreto, Galbraith, hEigeartaigh, and Scott, following the earlier work of Duursma and Lee [36]. Like most pairings, it uses a variant of Miller's algorithm to evaluate pairings. Its main feature, however, is that the η_T pairing requires only half the number of iterations of the Miller's loop compared with other pairings (Line 4, Algorithm 3 of [35]).

η_T spends most of its time performing extension field multiplications. Our code uses binary fields ($\mathbb{F}_{2^m}$), and the supersingular curve $y^2 + y = x^3 + x^2$, which has an *embedding degree* of four. In other words, this means that our implementation spends most of its time carrying out multiplications in $\mathbb{F}_{2^{4\times 271}}$, the quartic extension field. Next we will describe how TinyPBC computes binary field multiplication.

d) Finite Field and Big Number Arithmetic: The fastest known algorithms for multiplication on $\mathbb{F}_{2^m}$ first multiply the field elements as polynomials and then reduce the result modulo an irreducible polynomial $f(x)$. For the particular binary field $\mathbb{F}_{2^{271}}$, we have selected the pentanomial $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$, given in [37], which leads to a faster field square-root algorithm on a computer platform with word length of 8 or 16 bits.

Multiplication in our case is carried out using the *López-Dahab* (LD) method [38]. To compute the product ab, LD requires a look-up table for storing the product of polynomials of small degree by the operand b. We have used a look-up table of 16 polynomials; i.e., we process four bits in each iteration. For $b = (B[t-1], B[t-2], \ldots, B[0])$, the look-up table is defined as the matrix $T[i][j] = (i \cdot b)[j], i = 0, 1, ..., 15$ and $j = 0, \ldots, t$. In our code, T is computed by columns, i.e., in the following order: $T[i][t], T[i][t-1], \ldots, T[i][0]$ for $(i = 0, ..., 15)$.

Finite field multiplication is the most important operation, but not the only one that needs to be fast for efficiently computing pairings. To carry out other group and finite field operations as well as big number arithmetic, TinyPBC relies on MIRACL, a publicly available library written in C. The library has been used as basis for numerous works on efficient implementation. Also MIRACL already contains a highly efficient code for the η_T pairing. Running the code in resource-constrained nodes, such as those that use the ATmega128L, however, is not straight forward and thus adaptations have been made in order to fit MIRACL into the platform. In particular RAM usage must be minimized.

e) Optimization: To speed up the binary field multiplication (polynomial of size 34 bytes – as required to store 271 bits), we have come up with an implementation of LD that uses Karatsuba's method [39], another multiplication technique, of the depth-1. In other words, we use the LD method to multiply three polynomial of

[4] ID-NIKDS also requires hashing, but that can be efficiently computed in the ATmega128L [34].

size 17 bytes. In this case LD's main loop given by

```
for(i=0;i<17;i++){
    u = a[i]>>4;
    for(j=0;j<18;j++)
        c[i+j]^= T[u][j];
}
LSHIFT4(c); /* left shift c 4 bits*/
for(i=0; i<17; i++){
    u = a[i] & 0xf;
    for(j=0; j<18; j++)
        c[i+j]^= T[u][j];
}
```

was replaced by a code that uses a software pipeline technique and processes 8 bits in each iteration, namely

```
for(i=0;i<17;i++){
    u0 = a[i]&0xf; u1 = a[i]>>4;
    s0 = T[u0][0]; s1 = T[u1][0];
    c[i]^= s0^(s1<<4);
    s0 = T[u0][1]; s2 = T[u1][1];
    for(j=2;j<18;j++){
        c[i+j-1]^= s0^(s2<<4)^(s1>>4);
        s2 = s1; s0 = T[u0][j];
        s2 = T[u1][j];
    }
    c[i+17]^= s0^(s2<<4)^(s1>>4);
}.
```

Our improved version reduces the number of stores (for c) and it was observed that saves cycles (5.6%) when compared to the original LD algorithm. Note that the optimization have worked out for the GCC, but the result could have been different if were we using another compiler. Finally, it is worth noting that we have not used assembly and therefore our code is portable to other platforms.

B. Performance

In this section we summarize performance numbers. Figures are based on the avr-gcc compiler using optimization level -O3 in modules with time critical functions.

TinyPBC takes only 5.45s (Table I) to compute pairings on ATmega128L. That is, it requires less than half the amount of time of the quickest previous result [40], which takes 10.96s to compute pairings. This is mainly due to our faster binary field multiplication. The time required to compute binary field multiplication with our LD implementation, averaged over 20 trials, is just 4,019.46μs (Table I), with a standard deviation of 1.82μs. This is 44% faster than Karatsuba's method, which was employed in [40]. The result is particularly interesting because it contrasts sharply with results presented in [41], which claims that Karatsuba's is the most appropriate method for embedded devices.

Time	
Multiplication	**Pairing**
4,019.46μs	5.45s

TABLE I

TIME COSTS TO EVALUATE BINARY FIELD MULTIPLICATION AND THE η_T PAIRING ON ATMEGA128L USING TINYPBC.

C. Storage

Table II summarizes storage requirements of TinyPBC. The requirements for stack and static RAM as well as ROM are 2,687, 368 and, 47,948 bytes, respectively (Table II). Note that our approach allocates virtually all the RAM from the stack, which means that once the pairing is computed the memory is available for other operations. Plus, because the η_T pairing does not benefit from precomputation, ROM is saved.

Storage (bytes)		
Stack	**RAM**	**ROM**
2,867	368	47,948

TABLE II

MEMORY COSTS TO EVALUATE THE η_T PAIRING ON ATMEGA128L USING TINYPBC.

Besides the cryptographic code, a node needs to store its private key and public parameters in order to run ID-NIKDS. Public parameters are part of the specification of the pairing and they are already taken into consideration in our code. A private key, on the other hand, requires a point on $E(\mathbb{F}_{2^{271}})$. That is, an elliptic curve point that in turn is represented by coordinates (x, y) from the field $\mathbb{F}_{2^{271}}$, 271-bits each. Given x and a single bit of y, however, a node itself can easily derive y. So, in addition to the cryptographic code, a node must be loaded with a 272-bit private key, i.e., an overhead of only 34 bytes.

VI. RELATED WORK

The number of studies specifically targeted to secure WSNs has grown significantly. Due to space constraints, we first provide a sample of studies based on symmetric cryptosystems, and then focus on those targeted to efficient implementation of PKC on sensor nodes.

Many security proposals for WSNs (e.g., [13], [14], [15], [16], [17], [18], [19], [20], [22], [21], [23], [24]) have focused on efficient key management of symmetric encryption schemes. Perrig *et al.* [13] proposed SPINS, a suite of efficient symmetric key based security building blocks. Eschenauer *et al.* [14] looked at random key pre-distribution schemes, which provoked a large number of follow-on studies [19]. In [15] Zhu *et al.* proposed LEAP, a rather efficient scheme based on local distribution of secret keys among neighboring nodes.

The studies specifically targeted to PKC have tried either to adjust conventional algorithms (e.g. RSA) to sensor nodes, or to employ more efficient techniques (e.g. ECC) in this resource-constrained environment. All the seminal papers of Watro *et al.* [25], Gura *et al.* [26], and Malan *et al.* [27] have targeted the ATmega128L. Watro *et al.* [25] proposed TinyPK. To perform key distribution, TinyPK assigns the efficient RSA public operations to nodes and the expensive RSA private operations to better equipped external parties. Gura *et al.* [26] reported results for ECC and RSA primitives on the ATmega128L and demonstrated convincingly that the former outperforms the latter. Their ECC implementation is based upon arithmetic in the prime finite field $\mathbb{F}_p$. Malan *et al.* [27] have presented the first ECC implementation over binary fields $\mathbb{F}_{2^m}$ for sensor nodes. They used a polynomial basis and presented results for the ECDH key exchange protocol.

In the literature there are works that make use of identities to distribute keys in WSNs. Some (e.g Carman *et al.* [12], Du *et al.* [21], and Liu *et al.* [42]) are based on the symmetric cryptosystems due to Blundo *et al.* [43] and Blom *et al.* [44]. These strategies, however, do not provide perfect resilience, as after a certain percentage of nodes have been compromised, the whole network would be compromised as well. Others (e.g [45], [46], [47]) have employed IBC from PBC. The works of Zhang *et al.* [45], Doyle *et al.*, [46] and Oliveira *et al.* [47] employ IBC to distribute keys between nodes. However, they all use interactive protocols and therefore nodes are required to exchange messages to agree on keys.

Software implementation of pairings has also been focus of research. Oliveira *et al.* [48] have come up with an implementation of the Tate pairing. TinyTate, as it is called, uses TinyECC [42] as the underlying library and also targets the ATmega128L. TinyTate, however, takes around 31s to compute pairings and its level of security, equivalent to RSA-512, is not appropriate for all applications. More recently, Szczechowiak *et al.* [40] have shown performance number for ECC operations as well as pairings over binary and primes fields. Their implementation of η_T uses the Karatsuba's multiplication method and takes 10.96s to be evaluated.

VII. Conclusion

In spite of intense research efforts, the achievement of security in WSNs using cryptography still requires solutions. On the other hand, the advent of PBC has enabled a wide range of new cryptographic solutions. In this work, we first have shown how security in WSNs can be bootstrapped using ID-NIKDS. Subsequently we have presented TinyPBC, to our knowledge, the most efficient implementation of PBC primitives for an 8-bit processor. TinyPBC is able to compute pairings, the most expensive primitive of PBC, in about 5.5s on ATmega128L and it is based on MIRACL, an open source library.

VIII. Acknowledgment

We would like to thank Kenneth G. Paterson and Diego F. Aranha, for their valuable comments on this work, and CAPES and FAPESP, which support author L. B. Oliveira (under grants 4630/06-8 and 05/00557-9, respectively).

References

[1] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *MobiCom'99*, 1999, pp. 263–270.

[2] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *Symposium on Cryptography and Information Security (SCIS'00)*, Jan 2000, pp. 26–28.

[3] A. Joux, "The weil and tate pairings as building blocks for public key cryptosystems," in *ANTS-V: the 5th Int'l Symposium on Algorithmic Number Theory*, 2002, pp. 20–32.

[4] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Trans. on Information Theory*, vol. 39, no. 5, pp. 1639–1646, 1993.

[5] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003, also appeared in CRYPTO '01.

[6] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO'84*. Springer-Verlag, 1984, pp. 47–53.

[7] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Sp. Issue on Sensor Network Applications and Protocols*, vol. 1, no. 2–3, pp. 293–315, 2003, also in 1st IEEE Int'l Workshop on Sensor Networks Protocols and Applications.

[8] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *2nd ACM SensSys*, Nov 2004, pp. 162–175.

[9] J. L. Hill and D. E. Culler, "Mica: A wireless platform for deeply embedded networks." *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.

[10] M. Scott, *MIRACL—A Multiprecision Integer and Rational Arithmetic C/C++ Library*, Shamus Software Ltd, Dublin, Ireland, 2003, available at http://indigo.ie/ mscott.

[11] B. Schneier, *Applied Cryptography*, 2nd ed. Wiley, 1996.

[12] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and approaches for distributed sensor network security," NAI Labs, Network Associates, Inc., Tech. Rep. 00-010, 2000.

[13] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002, also in MobiCom'01.

[14] L. Eschenauer and V. D. Gligor, "A key management scheme for distributed sensor networks," in *9th ACM conf. on Computer and communications security (CCS'02)*, 2002, pp. 41–47.

[15] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *10th ACM conference on Computer and communication security (CCS'03)*. ACM Press, 2003, pp. 62–72.

[16] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *1st ACM workshop on Sec. of ad hoc and sensor net. (SASN'03)*, 2003.

[17] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy (S&P'03)*, may 2003, pp. 197–213.

[18] R. Kannan, L. Ray, and A. Durresi, "Security-performance tradeoffs of inheritance based key predistribution for wireless sensor networks," in *1st European Workshop on Security in Wireless and Ad-Hoc Sensor Networks (ESAS'04)*, 2004.

[19] J. Hwang and Y. Kim, "Revisiting random key pre-distribution schemes for wireless sensor networks," in *2nd ACM workshop on Security of ad hoc and sensor networks*, 2004, pp. 43–52.

[20] S. A. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks." in *9th European Symposium on Research Computer Security (ESORICS'04)*, 2004, pp. 293–308.

[21] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key pre-distribution scheme for wireless sensor networks," *ACM Trans. on Info. and System Security*, vol. 8, no. 2, pp. 228–58, 2005, also in ACM CCS'03.

[22] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Trans. on Info. and System Security*, vol. 8, no. 1, pp. 41–77, 2005, also in ACM CCS'03.

[23] L. B. Oliveira, H. C. Wong, R. Dahab, and A. A. F. Loureiro, "On the design of secure protocols for hierarchical sensor networks," *International Journal of Security and Networks (IJSN)*, vol. 2, no. 3/4, pp. 216–227, 2007.

[24] L. B. Oliveira, A. Ferreira, M. A. Vilaça, H. C. Wong, M. Bern, R. Dahab, and A. A. F. Loureiro, "SecLEACH– on the security of clustered sensor networks," *Signal Process.*, vol. 87, no. 12, pp. 2882–2895, 2007.

[25] R. J. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology." in *2nd ACM Workshop on Security of ad hoc and Sensor Networks (SASN'04)*, 2004, pp. 59–64.

[26] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus." in *Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)*, 2004, pp. 119–132.

[27] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, 2004.

[28] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *6th ACM MobiHoc '05*, New York, 2005, pp. 58–67.

[29] K. G. Paterson, "Cryptography from pairings," in *Advances in Elliptic Curve Cryptography*, ser. London Mathematical Society Lecture Notes, I. F. Blake, G. Seroussi, and N. Smart, Eds. Cambridge Univ. Press, 2005, vol. 317, ch. X, pp. 215–251.

[30] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *8th IMA Int'l Conference on Cryptography and Coding*. Springer-Verlag, 2001, pp. 360–363.

[31] A. Joux, "A one round protocol for tripartite diffie-hellman." *J. Cryptology*, vol. 17, no. 4, pp. 263–276, 2004, also in ANTS'00.

[32] S. D. Galbraith, "Pairings," in *Advances in Elliptic Curve Cryptography*, ser. London Mathematical Society Lecture Notes, I. F. Blake, G. Seroussi, and N. Smart, Eds. Cambridge Univ. Press, 2005, vol. 317, ch. IX, pp. 183–213.

[33] S. Galbraith, K. Paterson, and N. Smart, "Pairings for cryptographers," Cryptology ePrint Archive, Report 2006/165, 2006.

[34] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes," in *ACM Int'l conf. on Wireless sensor networks and applications*, 2003, pp. 151–159.

[35] P. S. L. M. Barreto, S. Galbraith, C. O. hEigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," in *Designs Codes And Cryptography*, 2006.

[36] M. Duursma and H.-S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," in *9th ASIACRYPT'03*. Springer, 2003, pp. 111–123.

[37] M. Scott, "Optimal irreducible polynomials for GF(2^m) arithmetic," Cryptology ePrint Archive, Report 2007/192, 2007.

[38] J. López and R. Dahab, "High-speed software multiplication in GF(2^m)," in *Progress in Cryptology - INDOCRYPT'00*, 2000, pp. 203–212, lecture Notes in Computer Science.

[39] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Soviet Physics-Doklad (Engl. transl.)*, vol. 7, no. 7, pp. 595–596, 1963.

[40] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: Testing the limits of elliptic curve cryptography in sensor networks," in *European conference on Wireless Sensor Networks (EWSN'08)*, 2008, pp. 305–320.

[41] S. Bartolini, I. Branovic, R. Giorgi, and E. Martinelli, "Effects of instruction-set extensions on an embedded processor: a case study on elliptic curve cryptography over GF(2^m)," *IEEE Transactions on Computers*, 2007, to appear.

[42] A. Liu, P. Kampanakis, and P. Ning, "Tinyecc: Elliptic curve cryptography for sensor networks (ver. 0.3)," 2005, http://discovery.csc.ncsu.edu/software/TinyECC/.

[43] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *CRYPTO '92*, 1992, pp. 471–486.

[44] R. Blom, "An optimal class of symmetric key generation systems," in *EUROCRYPT' 84*, 1984, pp. 335–338.

[45] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing sensor networks with location-based keys," in *IEEE Wireless Communications and Networking Conference (WCNC'05)*, 2005.

[46] B. Doyle, S. Bell, A. F. Smeaton, K. McCusker, and N. O'Connor., "Security considerations and key negotiation techniques for power constrained sensor networks," *The Computer Journal*, vol. 49, no. 4, pp. 443–453, 2006.

[47] L. B. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A. A. F. Loureiro, "Identity-based encryption for sensor networks," in *5th IEEE Int'l Conference on Pervasive Computing and Communications Workshops (PERCOMW '07)*, 2007, pp. 290–294.

[48] L. B. Oliveira, D. Aranha, E. Morais, F. Daguano, J. López, and R. Dahab, "TinyTate: Computing the tate pairing in resource-constrained nodes," in *6th IEEE International Symposium on Network Computing and Applications*, July 2007, pp. 318–323.