Computer Communications xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks

Leonardo B. Oliveira [a,*], Diego F. Aranha [b], Conrado P.L. Gouvêa [b], Michael Scott [c], Danilo F. Câmara [b], Julio López [b], Ricardo Dahab [b]

[a] Faculty of Technology, UNICAMP, Limeira, SP, CEP 13484-332, Brazil
[b] Institute of Computing, UNICAMP, Campinas, SP, CEP 13083-970, Brazil
[c] School of Computing, Dublin City University, Glasnevin, Dublin 9, Ireland

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Key distribution
Sensor networks
Identity-based cryptography
Pairing-based cryptography
Efficient implementation

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, as they require the exchange and storage of large keys and certificates, which is expensive.

Using Pairing-Based Cryptography (PBC) protocols 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 8, 16 and 32-bit processors commonly found in sensor nodes. TinyPBC is able to compute pairings, the most expensive primitive of PBC, in 1.90 s on ATmega128L, 1.27 s on MSP430 and 0.14 s on PXA27x.

1. 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] is an emerging technology that allows a wide range of applications. Pairings have been attracting the interest of the international cryptography community because they enable the design of original cryptographic schemes and make well-known cryptographic protocols more efficient. Perhaps the main evidence of this is the realization of Identity-Based Encryption (IBE) [4] which in turn has facilitated complete schemes for Identity-Based Cryptography (IBC) [5].

In the context of 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 [6]. One simple idea to introduce minimal security is to fit each sensor node with the same cryptographic key to be used for all communications (e.g. [7]). 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 $ID \in \{0,\ldots,n-1\}$. 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. 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 WSN without any impact on existing nodes.

This scheme exists and in the area of Cryptography it is known as an *Identity-Based Non-Interactive Key Distribution Scheme*

* Corresponding author. Address: R. Paschoal Marmo, 1888 – CEP 13484-332 – Jd., Nova Itália – Limeira, SP, Brazil. Tel.: +55 19 2113 3368; fax: +55 19 2113 3339.
E-mail addresses: leob@ft.unicamp.br (L.B. Oliveira), dfaranha@ic.unicamp.br (D.F. Aranha), conradoplg@ic.unicamp.br (C.P.L. Gouvêa), mike@computing.dcu.ie (M. Scott), dfcamara@gmail.com (D.F. Câmara), jlopez@ic.unicamp.br (J. López), rdahab@ic.unicamp.br (R. Dahab).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.05.013

(ID-NIKDS) [2]. It is *Identity-Based* [8], 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. In fact "non-interactive" implies "Identity-based": In this setting the only information a node knows about another node is its identity. And it is a *Key Distribution Scheme*, because each node pair 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 – from where does each entity get its unique secret? 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 setup 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 should be 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 is 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 (PKC)? No, because in regular PKC there is no correlation between an individual's ID and their public key. 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 bootstrapping 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 is only 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 unique secret is unavoidable if authentication is a requirement.

We do not claim that a scheme like this is by itself sufficient for securing WSNs and 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 first 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 the 8, 16 and 32-bit processors found in sensor nodes. Performance figures are presented for the processors ATmega128L (the MICA2 and MICAZ node microcontroller [9]), MSP430 (the TelosB and Tmote Sky microcontroller [10]) and PXA271 (the Imote2 microcontroller [11]). TinyPBC is based on the RELIC cryptographic toolkit [12], which is a publicly available and open source library. 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 several sensor platforms; and
(3) show the best figures for the implementation of finite field arithmetic in these platforms.

The remainder of this work is organized as follows. In Section 2, we discuss the need for new security solutions in WSNs. We point out the synergy between IBC and WSNs in Section 3. In Section 4 we show how ID-NIKDS can bootstrap security in WSNs. Implementation and results are presented in Section 5. Finally, we discuss related work and conclude in Sections 6 and 7, respectively.

2. 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 industrial and other 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 single node should not impact the network as whole. (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.

In WSNs, security is typically bootstrapped using key distribution schemes. Most of standard key distribution schemes in the security literature [13], 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. [14] for a good introduction to key distribution in WSNs.)

Symmetric key-based distribution schemes have been specifically designed for WSNs (e.g. [15–26]). 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, while not providing an ideal level of either. Further, most schemes rely on some sort of interaction between nodes so that they can agree on keys.

Subsequently, it has been shown that methods of Public Key Cryptography are feasible in WSNs [27–29]. 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. [30]), often ends up in overhead – which is especially ill-suited to WSNs.

As we will show in Section 4, by using ID-NIKDS we are able to resolve these security issues.

3. Synergy between IBC and WSNs

PBC has paved the way for a new wide range of cryptographic protocols and applications [31]. It has also allowed many long-standing open problems to be solved elegantly. Perhaps the most impressive among those applications is IBE [4], which in turn has allowed complete IBC schemes [5].[2]

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

[2] Note that other methods of implementing IBE exist (e.g. [32]).

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

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, IBC also faces additional drawbacks. In particular, it requires a Private Key Generator (PKG), a trusted 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, the PKG must be an entity that is unconditionally trusted by all network users. Such an entity, however, cannot always be easily identified in many scenarios.

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

Another IBC 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 – 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, private keys can be loaded into nodes during the pre-deployment stage.

4. Authenticated identity-based non-interactive key distribution in sensor networks

The notion of Identity-Based Cryptography dates back from Shamir's original work [5], but it has only become practical with the advent of PBC [2,4,33]. 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 (Section 4.1), and show how to setup IBC schemes in the WSN context (Section 4.2), and finally show how ID-NIKDS can be used so that pairs of nodes can establish common secret keys (Section 4.3).

4.1. Pairings: definition

Bilinear pairings – or pairings for short – were first used in the context of cryptanalysis [34], but their pioneering use in cryptosystems is due to the works of Sakai et al. [2] and Joux [33]. In what follows, let $E/\mathbb{F}_q$, $E(\mathbb{F}_q)$ 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, that is the number of points on the curve.

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.

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

$$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T,$$

with additional properties, the most important of which, in cryptographic constructions, is bilinearity; namely:

$$\forall P, Q \in \mathbb{G}, \quad \text{and} \quad \forall a, b \in \mathbb{Z}^*, \quad \text{we have}$$
$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab}.$$

For practical implementations, the group $\mathbb{G}$ is chosen as a set of points on certain elliptic curves and the group $\mathbb{G}_T$ is a multiplicative subgroup of a finite extension field. For more on pairing definitions, see, for instance, Galbraith [35].

Here we are using the definition of a *Type 1* or *symmetric* pairing (in the sense of Galbraith et al. [36]), 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 protocols (pairings of *Types 2* and *3* do not provide, at once, both properties). However, realization of the solution is possible with *Type 3* pairings and some added complexity [37]. Pairing computation in sensor nodes at a sufficient security level for WSNs also seems more efficient in the symmetric setting [37].

4.2. Setup

To start up an IBC scheme, the PKG first needs to generate and distribute private keys and public parameters. Broadly speaking, this procedure can be accomplished as follows in WSNs. First, 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; and (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 output 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.

4.3. 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 2 (i.e., perfect connectivity and resilience), an ideal key agreement scheme for WSNs should also keep the number of exchanged messages to a minimum.

With the advent of PBC, however, a method of accomplishing this has become 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 et al. ID-NIKDS [2], can be employed to achieve such a goal. We assume that the setup protocol shown in Section 4.2 has been already carried out.

Suppose two nodes A and B that know each other's IDs have to agree on a secret key. Recall from Section 4.2 that nodes' A and B private keys are $S_A = [s]P_A$ and $S_B = [s]P_B$, respectively. Consequently, by bilinearity (Section 4.1) we have

$$e(S_A, P_B) = e([s]P_A, P_B) = e(P_A, P_B)^s = e(P_A, [s]P_B) = e(P_A, S_B) = e(S_B, P_A).$$

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

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

(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 [31].) Additionally, A knows that only B – and the BS, a Trusted Authority – possess S_B and vice versa, and consequently the protocol is authenticated.

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

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

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, observe that we assume that nodes already know each other's IDs, a reasonable assumption in WSNs since in these networks nodes already need to get to know their neighbors' IDs to exchange ordinary information.

5. Evaluation

The utilization of pairings to implement security in WSNs is quite complex. For an 80-bit security level (RSA-1024 equivalent), PBC works with 1024-bit numbers – as opposed to conventional Elliptic Curve Cryptography (ECC), which works with 160-bit numbers only. In this section, we assess the costs incurred by PBC on several representatives of the sensor platform spectrum: a resource-constrained MICAZ node, a (TelosB) Tmote Sky node and a powerful Imote2 node. The MICA platform features an 8-bit ATmega128 microcontroller, the Tmote Sky node employs a 16-bit TI MSP430 microcontroller and the Imote2 platform has a 32-bit ARM XScale PXA27x microcontroller.

5.1. 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 Tate pairing (realized in the form of the η_T [39] pairing – pronounced "eta-t" – over supersingular binary curves and a variant of the Tate pairing over prime fields [37]) for resource-constrained nodes. The source code is available at URL http://sites.google.com/site/tinypbc/.

5.1.1. Security requirements

To meet efficiency constraints, security requirements in WSNs are often relaxed. For example, some (e.g. [15]) have adopted a 64-bit security level. We adopted a more conservative posture and thus used an 80-bit security level, as recommended by NIST.

5.1.2. Pairing

The implementation comprised the η_T [39] pairing which is defined over binary fields. This pairing is possibly the fastest known pairing at this security level [37]. It was proposed by Barreto, Galbraith, Ó hÉigeartaigh, and Scott, following earlier work of Duursma and Lee [40]. 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 [39]).

Our sofware implementation of the η_T pairing is for binary fields ($\mathbb{F}_{2^{271}}$). We selected the supersingular curve $y^2 + y = x^3 + x$, which has an *embedding degree* of four. In this case, the execution of the η_T pairing spends most of its time performing field multiplications in $\mathbb{F}_{2^{4\times 271}}$, the quartic extension field.

5.2. ATmega128 8-bit processor

The MICAz Mote sensor node is equipped with an ATmega128 8-bit processor clocked at 7.3728 MHz. The program code is stored in a 128 KB EEPROM chip and data memory is provided by a 4 KB RAM chip [9]. The ATmega128 processor is a typical RISC architecture with 32 registers, but six of them are special pointer registers. Since at least one register is needed to store temporary results or data loaded from memory, 25 registers are generally available for arithmetic. The instruction set is also reduced, as only 1-bit shift/rotate instructions are natively supported. Bitwise shifts by arbitrary amounts can then be implemented with combinations of shift/rotate instructions and other instructions. In particular, shifts by 7 bits can be implemented very efficiently with the instructions `bld`/`bst` for loading/storing individual register bits from/to a processor flag. The processor pipeline has two stages and memory instructions always cause pipeline stalls. Arithmetic instructions with register operands cost 1 cycle and memory instructions or memory addressing cost 2 processing cycles [41].

5.2.1. Field representation

The elements of the binary field are represented using a polynomial basis. For the particular binary field $\mathbb{F}_{2^{271}}$, we have selected the square-root friendly [42] pentanomial $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$, given in [43]. This pentanomial has two important features: modular reduction by $f(z)$ only requires shifts by 1 bit or 7 bits which are fast in this platform; square-root extraction does not require shifts in processors with word length of 8 or 16 bits. In software, a field element $a(z)$ is stored as an array of $n = 34$ bytes.

5.2.2. Squaring

Given $a(z) \in \mathbb{F}_{2^{271}}$, the binary representation of $a(z)^2$ can be computed by inserting a "0" bit between each pair of successive bits of the binary representation of $a(z)$. This can be accelerated by introducing a small 16-byte lookup table which stores in memory the square of all 4-bit polynomials. In platforms with expensive access to memory, redundant memory accesses can be avoided by implementing squaring in two steps. The first one computes the square of the lower half of the digit vector using a conventional 4-bit expansion table. The second step computes the square of the higher half combined with modular reduction. This way, values already loaded into registers can be reduced immediately.

5.2.3. Multiplication

Multiplication is a performance-critical operation and was implemented with the López–Dahab algorithm [44] using a window size of $t = 4$ bits (Algorithm 1). In order to avoid redundant memory accesses, the intermediate digit vector is stored inside a rotating register window as in Algorithm 2. This optimization alone reduces the number of read instructions by half and the number of write instructions by a quadratic factor, compared with a standard implementation of the algorithm. Since this register window would require 35 registers and only 25 are available for arithmetic, the accumulation in the register window was divided in different blocks in a multistep fashion and each block processed with a different rotating register window. A slight overhead is introduced between the processing of consecutive blocks because some registers must be written into memory and freed before they can be used in a new rotating register window. Some additional implemented optimizations are: storing the results of the first phase of the algorithm already shifted; and the embedding of modular reduction at the end of the multiplication algorithm, again by making use of results already stored in registers and avoiding redundant memory accesses.

[4] ID-NIKDS also requires hashing, but that can be efficiently computed in sensor nodes [38].

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

Algorithm 1: López–Dahab multiplication in $\mathbb{F}_{2^m}$ [44].

Input: $a(z) = a[0..n-1]$, $b(z) = b[0..n-1]$.
Output: $c(z) = c[0..2n-1]$.
1: Compute $T(u) = u(z)b(z)$ for all polynomials $u(z)$ of degree lower than t.
2: $c[0...2n-1] \leftarrow 0$
3: **for** $k \leftarrow 0$ to $n-1$
4: $u \leftarrow a[k] \gg t$ **do**
5: **for** $j \leftarrow 0$ to n **do**
6: $c[j+k] \leftarrow c[j+k] \oplus T(u)[j]$
7: **end for**
8: **end for**
9: $c(z) \leftarrow c(z)z^t$
10: **for** $k \leftarrow 0$ to $n-1$
11: $u \leftarrow a[k] \bmod 2^t$
12: **for** $j \leftarrow 0$ to n
13: $c[j+k] \leftarrow c[j+k] \oplus T(u)[j]$
14: **end for**
15: **end for**
16: **return** c

Algorithm 2: Proposed optimization for multiplication in $\mathbb{F}_{2^m}$ using $n+1$ 8-bit registers.

Input: $a(z) = a[0..n-1]$, $b(z) = b[0..n-1]$.
Output: $c(z) = c[0..2n-1]$.
Note: v_i denotes the vector of $n+1$ registers $(r_{i-1},\ldots,r_0,r_n,\ldots,r_i)$.
1: Compute $T(u) = u(z)b(z)$ for all polynomials $u(z)$ of degree lower than 4.
2: Let u_i be the 4 most significant bits of $a[i]$.
3: $v_0 \leftarrow T(u_0)$, $c[0] \leftarrow r_0$
4: $v_1 \leftarrow v_1 \oplus T(u_1)$, $c[1] \leftarrow r_1$
5: $\cdots$
6: $v_{n-1} \leftarrow v_{n-1} \oplus T(u_{n-1})$, $c[n-1] \leftarrow r_{n-1}$
7: $c \leftarrow ((r_{n-2},\ldots,r_0,r_n)\|(c[n-1],\ldots,c[0])) \ll 4$
8: Let u_i be the 4 least significant bits of $a[i]$.
9: $v_0 \leftarrow T(u_0)$, $c[0] \leftarrow c[0] \oplus r_0$
10: $\cdots$
11: $v_{n-1} \leftarrow v_{n-1} \oplus T(u_{n-1})$, $c[n-1] \leftarrow c[n-1] \oplus r_{n-1}$
12: $c[n...2n-1] \leftarrow c[n...2n-1] \oplus (r_{n-2},\ldots,r_0,r_n)$
13: **return** c

5.2.4. Square-root

Square-root extraction was implemented according to the approach due to Fong et al. [45]. This approach requires a splitting step for concatenating the coefficients with even or odd indexes of a field element. This is commonly implemented using a lookup table (as in [37]), but the microcontroller support for bit load/store operations allowed a particularly fast implementation without table lookups using simple bit manipulation.

5.3. MSP430 16-bit processor

The Tmote Sky sensor node is equipped with an MSP430F1611 16-bit processor clocked at 8 MHz. It contains 48 KB of program flash memory and 10 KB of RAM. The MSP430 family provides 12 general-purpose registers and a small instruction set with 27 instructions including 1-bit-only shifts. In particular, 15-bit shifts can be implemented with the left-shift/rotate-through-carry instructions. Operands may be located in registers or in memory. Since there is no cache, determining the number of cycles taken by each instruction is simple (with a few exceptions): one cycle to fetch the instruction, one cycle to fetch each offset word (if any), one cycle for each memory read and two cycles for each memory write. Small constants (−1, 0, 1, 2, 4 and 8) are generated by using some special registers and do not require offset words when used. Four addressing modes are available: direct (from registers), indirect (memory address stored in a register), indexed (address stored in a register, plus an offset) and indirect with post increment (which automatically increments the contents of the register holding the address).

5.3.1. Field representation

The same pentanomial $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$ used in the ATmega128 processor was chosen, since the reduction by $f(z)$ can be computed using 1-bit and 15-bit shifts only, which are cheap in this platform. Additionally, square-root extraction does not require any shifts. In software, a field element $a(z)$ is stored as an array of $n = 17$ 16-bit digits.

5.3.2. Arithmetic

Field arithmetic was implemented similarly to the ATmega128 processor. Squaring was implemented using a 512-byte lookup table storing the square of all 8-bit polynomials in ROM. Multiplication was implemented by a straightforward adaptation of Algorithm 2 for 16-bit processors processing the consecutive accumulations in two distinct blocks and employing a rotating window with 8 registers. For comparison, the Karatsuba method was also implemented as described in [37]. Some simple but effective optimizations used were taking advantage of the indirect with post increment addressing mode, which saves one cycle on each read from the precomputation table; and computing the required 4-bit shifts in registers, dividing the intermediate result into four blocks.

5.4. XScale PXA27x 32-bit processor

ARM is an open RISC processor architecture that is the most widely used in 16/32-bit embedded RISC solutions. Besides typical RISC architecture features, ARM includes: an optional embedding of a shift operation in every data-processing instruction executed by the Arithmetic Logic Unit (ALU); auto-increment/decrement addressing modes to optimize program loops; and conditional execution of all instructions to maximize execution throughput.

From the user mode (unpriviledged code) point of view, the ARM has 16 general-purpose 32-bit registers. Two of these registers have special roles: register 15 is the Program Counter (PC) and register 14 is the Link Register (LR) that holds the address of the next instruction after a Branch and Link (BL) instruction used to call a subroutine. Software implementation normally uses R13 as the Stack Pointer (SP).

The Intel XScale PXA family of processors is an implementation of the 5th generation of the ARM architecture without the floating point instructions. With a design tailored for wireless and portable multimedia devices, the PXA family focuses on balancing processing power and battery usage. The target processor used in this work is the PXA271, a 32-bit ARMv5TE with 32 KB data cache and 32 KB instruction cache. This family of processors also features a Wireless MMX (WMMX) coprocessor which executes vector instructions in 64-bit registers. The WMMX instruction set supports 43 SIMD instructions for orthogonal manipulation of 16 architectural registers that can be treated as arrays of 32-bit words, four 16-bit halfwords or eight 8-bit bytes.

5.4.1. Field representation

In the ARM implementation we selected the square-root friendly trinomial $f(x) = x^{271} + x^{201} + 1$ given in [37]. Shift instructions are flexible in this platform and, under these circumstances, this trinomial allows fast implementations of modular reduction

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

and square-root extraction. In software, a field element $a(z)$ is stored as an array of $n = 9$ 32-bit words.

5.4.2. Arithmetic

Similarly to the ATmega128 processor, we optimized the implementation with a lookup table in the squaring operations and employed a rotating register window as in Algorithm 2 in the multiplication operation. We were able to reserve 9 registers to the accumulator and the multiplication did not need to be divided as in the 8-bit case. Since a bitwise shift is free in any data-processing instruction, we embedded most shifts of the arithmetic in the binary addition instructions (logic XOR operation).

5.5. Performance

In this section we summarize performance numbers. High-level algorithms like pairing computation and extension field arithmetic were implemented in C, while finite field arithmetic was implemented in Assembly to allow fine-grained resource allocation and to avoid inefficiencies introduced by the compiler. Figures are based on the GCC 4.1.2 compiler with optimization level `-O2` and loop unrolling (GCC 3.2.3 on the MSP430). The timings were measured with the software AVR Studio 4.14 [46], the cycle-accurate simulator MSPsim [47] and on a PXA27x Mainstone evaluation board.

Table 1
Time costs to evaluate finite field arithmetic and the η_T pairing at the 80-bit security level on MICA2/MICAz platform (7.3728 MHz ATmega128L) using TinyPBC.

Algorithm	TinyPBC Cycles	Szczechowiak et al. [37] Cycles	Comparison Improvement (%)
Squaring	1439	1581	9.0
Square-root	1182	1730	31.8
Multiplication	11727	13557	13.5
Pairing	14×10^6 (1.90 s)	19.6×10^6	28.6

Table 2
Time costs to evaluate finite field arithmetic and the η_T pairing at the 80-bit security level on TelosB/Tmote Sky platform (8 MHz MSP430) using TinyPBC.

Algorithm	TinyPBC Cycles	Szczechowiak et al. [37] Cycles	Comparison Improvement (%)
Squaring	889	1363	34.8
Square-root	769	1644	53.2
Mult. (Karatsuba)	9647	10147	4.0
Multiplication	8706	–	13.3
Pairing (Karatsuba)	11.3×10^6 (1.38 s)	14.1×10^6	19.8
Pairing	10.4×10^6 (1.27 s)	–	26.3

Table 3
Time costs to evaluate finite field arithmetic and the η_T pairing at the 80-bit security level on Imote2 platform (13 MHz PXA27x) using TinyPBC.

Algorithm	TinyPBC Cycles	Szczechowiak et al. [37] Cycles	Comparison Improvement (%)
Squaring	187	499	62.5
Square-root	185	546	66.1
Multiplication	2025	4926	58.9
Mult. (wMMX)	1411	–	71.3
Pairing	2.45×10^6 (0.19 s)	6×10^6	59.2
Pairing (WMMX)	1.81×10^6 (0.14 s)	–	69.8

TinyPBC takes only 1.90 s to compute pairings on ATmega128L, 1.27 s on the Tmote Sky and 0.14 s on a 13 MHz Imote2 (Tables 1–3). That is, it requires 28.6%, 26.3% and 69.8% less than the time of the fastest previous results [37], which take 2.66 s, 1.71 s and 0.46 s to compute pairings on these platforms. This is mainly due to our faster finite field multiplication and to some further algorithmic improvements proposed in [48]. The time required to compute binary field multiplication with our LD implementation, averaged over 1000 trials, is only 11727 cycles in Table 1, 8706 cycles in Table 2 and 1411 cycles in Table 3. For the ATmega128 platform, this is 13.5% faster than Karatsuba's method, which was employed in [37]. This result is particularly interesting because it contrasts sharply with results presented in [49], which claims that Karatsuba's is the most appropriate method for embedded devices. For the Tmote Sky, the LD method using blocks resulted in a 13.3% speed improvement; but Karatsuba may be used to provide a speed/space trade-off, raising the pairing computation time from 1.27 s to 1.38 s. For the Imote2 platform, the WMMX instruction set included in the XScale family of processors provides further speedups and we achieve an improvement of 71.3% in our implementation of multiplication.

5.6. Storage

Table 4 summarizes the storage requirements for TinyPBC. The requirements for stack and static memory and program size for all platforms are presented. Note that our approach allocates virtually all the RAM from the stack, which means that once the pairing is computed, memory is available for other operations. The MSP430 has the smallest ROM space available of the three platforms; for that reason, the Karatsuba multiplication method can be used in order to save 6 KB of ROM, increasing the pairing computation time by 0.11 s in return. We have selected RELIC [12] as our implementation framework because this library was specifically built for memory-contrained devices, providing several configuration options for reducing program size.

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 an elliptic curve; that is, an elliptic curve point that, in turn, is represented by coordinates (x,y) from a finite field with 271-bit elements. Given x and a single bit of y, however, one can easily derive y. So, in addition to the cryptographic code, a node must be loaded with a private key of 272 bits, i.e., an overhead of only 34 bytes.

6. Related work

The number of studies specifically targeted to secure WSNs has grown significantly. Due to space constraints, we first provide a sam-

Table 4
Memory costs to evaluate the pairings on the target platforms using the fastest finite field implementations. The RAM column refers to global variables.

	Storage (KB)		
Processor	Stack	RAM	ROM
ATmega128L	3.1	0.5	37.9
MSP430	2.8	0.5	30.1
MSP430 (Karatsuba)	2.4	0.5	24.0
PXA27x	8.0	4.7	53.5

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

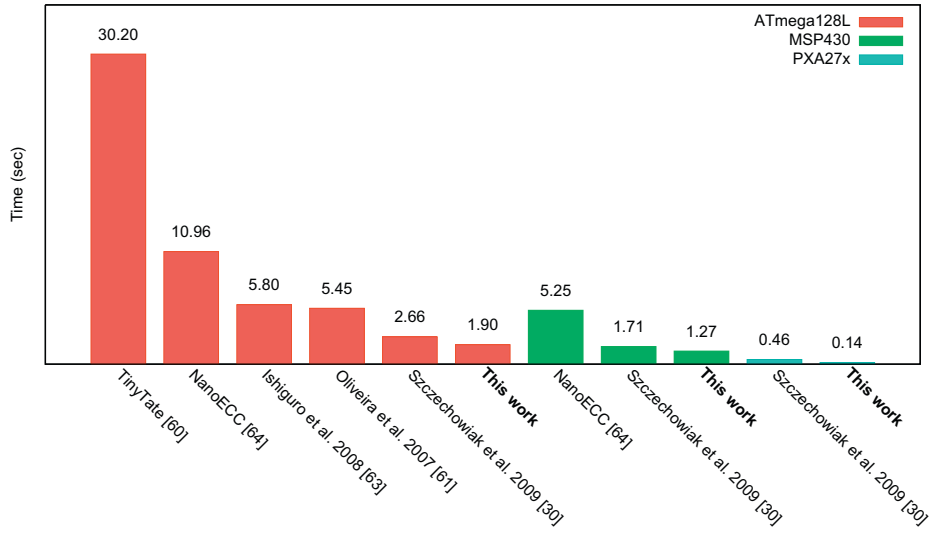

Fig. 1. Timings for pairing computation in several sensor platforms.

ple 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. [15–22,24,23,25,26]) have focused on efficient key management of symmetric encryption schemes. Perrig et al. [15] proposed SPINS, a suite of efficient symmetric key-based security building blocks. Eschenauer and Gligor [16] looked at random key predistribution schemes, which opened the way to a large number of follow-up works [21]. In [17] 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. [27], Gura et al. [28], and Malan et al. [29] have targeted the ATmega128L. Watro et al. [27] 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. [28] 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. [29] have all 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. [14,23,50]) are based on the symmetric cryptosystems due to Blundo et al. [51] and Blom [52]. These strategies, however, do not provide perfect resilience, as after a certain percentage of nodes have been compromised, the whole network can be compromised as well. Others (e.g. [53–55]) have employed IBC from PBC. The works of Zhang et al. [53], Doyle et al., [54] and Oliveira et al. [55] 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. Before demonstrating the efficiency of PBC on sensor nodes, we first showed its feasibility [56] with an implementation of the Tate pairing. TinyTate, as it is called, uses TinyECC [50] as the underlying library and also targets the ATmega128L. TinyTate, however, takes around 31 s to compute pairings and its level of security, equivalent to RSA-512, is not appropriate for all applications. We subsequently – in a previous version of TinyPBC [57] – improved these figures by employing the MIRACL cryptographic library [58] together with the López–Dahab binary field multiplication method [44]. By using TinyPBC, we were able to compute the η_T pairing in only 5.45 s on the ATmega128L. The same pairing has been implemented in the work of Ishiguro et al. [59] as well. Their work uses ternary fields and evaluates pairings in 5.79 s. Also using MIRACL, Szczechowiak et al. [60] have shown performance numbers for ECC operations as well as pairings over binary and primes fields. Their implementation of η_T uses the Karatsuba multiplication method and takes 10.96 s to be evaluated on the ATmega128L and 5.25 s on the MSP430. Further, by translating critical part of code to Assembly and then by carefully manipulating registers, Szczechowiak et al. [37] managed to reduce those times to only 2.66 s and 1.71 s, respectively. Fig. 1 presents a comparison between the execution times obtained by this work compared to the related work discussed in this section.

More recently, Oliveira et al. [61] have shown how short signatures [62] from pairings can be used to authenticate sensors in a shared WSN scenario and Galindo et al. [63] have used TinyPBC to make explicit the benefits of using PBC to solve the key distribution problem in Underwater Wireless Sensor Networks (UWSNs).

7. Conclusion

In spite of intense research efforts, achieving security in WSNs using cryptography is still a challenging problem. 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 8, 16 and 32-bit processors found in wireless sensor nodes. TinyPBC is able to compute pairings, the most expensive primitive of PBC, in 1.90 s on ATmega128L, 1.27 s on MSP430 and 0.14 s on PXA27x and it is based on RELIC [12], an open source cryptographic library.

Acknowledgements

The authors thank Kenneth G. Paterson, Paulo S. L. M. Barreto, Piotr Szczechowiak and the anonymous reviewers for their

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

valuable comments on this work; and CAPES and FAPESP, which support authors L. B. Oliveira and D. F. Aranha (under grants 4630/06–8, 05/00557–9; and 07/06950–0 respectively).

References

[1] D. Estrin, R. Govindan, J.S. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, in: K.G. Shin, Y. Zhang, R. Bagrodia, R. Govindan (Eds.), 15th Annual International Conference on Mobile Computing and Networking (MOBICOM'99), ACM Press, Seattle, WA, USA, 1999, pp. 263–270.

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

[3] A. Joux, The weil and tate pairings as building blocks for public key cryptosystems, in: C. Fieker, D.R. Kohel (Eds.), The Fifth International Symposium on Algorithmic Number Theory (ANTS-V), LNCS, Vol. 2369, Springer, 2002, pp. 20–32.

[4] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, SIAM Journal of Computing 32 (3) (2003) 586–615. also appeared in CRYPTO'01.

[5] A. Shamir, Identity-based cryptosystems and signature schemes, in: G.R. Blakley, D. Chaum (Eds.), Fourth Annual International Cryptology Conference (CRYPTO'84), LNCS, Vol. 196, Springer, 1984, pp. 47–53.

[6] C. Karlof, D. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols 1 (2–3) (2003) 293–315. also appeared in SNPA'03.

[7] C. Karlof, N. Sastry, D. Wagner, TinySec: a link layer security architecture for wireless sensor networks, in: J.A. Stankovic, A. Arora, R. Govindan (Eds.), Second International Conference on Embedded Networked Sensor Systems (SenSys'04), ACM Press, 2004, pp. 162–175.

[8] C. Boyd, K.-K.R. Choo, Security of two-party identity-based key agreement, in: E. Dawson, S. Vaudenay (Eds.), First International Conference on Cryptology in Malaysia (MYCRYPT'05), LNCS, Vol. 3715, Springer, 2005, pp. 229–243.

[9] J.L. Hill, D.E. Culler, Mica: A wireless platform for deeply embedded networks, IEEE Micro 22 (6) (2002) 12–24.

[10] J. Polastre, R. Szewczyk, D. Culler, Telos: enabling ultra-low power wireless research, in: Fourth International Symposium on Information processing in Sensor Networks (IPSN'05), IEEE Press, Piscataway, NJ, USA, 2005, pp. 364–369.

[11] L. Nachman, R. Kling, R. Adler, J. Huang, V. Hummel, The intel mote platform: a bluetooth-based sensor network for industrial monitoring, in: Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05), IEEE Press, 2005, pp. 437–442.

[12] D.F. Aranha, RELIC Cryptographic Toolkit, <http://code.google.com/p/relic-toolkit>, 2009.

[13] A. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

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

[15] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor networks, Wireless Networks 8 (5) (2002) 521–534. also appeared in MOBICOM'01.

[16] L. Eschenauer, V.D. Gligor, A key management scheme for distributed sensor networks, in: V. Atluri (Ed.), Ninth ACM Conference on Computer and Communications Security (CCS'02), ACM Press, 2002, pp. 41–47.

[17] S. Zhu, S. Setia, 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.

[18] R.D. Pietro, L.V. Mancini, A. Mei, Random key-assignment for secure wireless sensor networks, in: S. Setia, V. Swarup (Eds.), First ACM Workshop on Security of ad hoc and Sensor Networks (SASN'03), ACM Press, 2003, pp. 62–71.

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

[20] R. Kannan, L. Ray, A. Durresi, Security-performance trade-offs of inheritance based key predistribution for wireless sensor networks, in: First European Workshop on Security in Wireless and ad hoc Sensor Networks (ESAS'04), 2004.

[21] J. Hwang, Y. Kim, Revisiting random key predistribution schemes for wireless sensor networks, in: S. Setia, V. Swarup (Eds.), Second ACM Workshop on Security of ad hoc and Sensor Networks (SASN'04), 2004, pp. 43–52.

[22] S.A. Çamtepe, B. Yener, Combinatorial design of key distribution mechanisms for wireless sensor networks, in: P. Samarati, P.Y.A. Ryan, D. Gollmann, R. Molva (Eds.), Ninth European Symposium on Research Computer Security (ESORICS'04), LNCS, Vol. 3193, Springer, 2004, pp. 293–308.

[23] W. Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, A. Khalili, A pairwise key predistribution scheme for wireless sensor networks, ACM Transactions on Information and System Security 8 (2) (2005) 228–258. also appeared in ACM CCS'03.

[24] D. Liu, P. Ning, R. Li, Establishing pairwise keys in distributed sensor networks, ACM Transactions on Information and System Security 8 (1) (2005) 41–77. also appeared in ACM CCS'03.

[25] L.B. Oliveira, H.C. Wong, R. Dahab, A.A.F. Loureiro, On the design of secure protocols for hierarchical sensor networks, International Journal of Security and Networks (IJSN) 2 (3/4) (2007) 216–227.

[26] L.B. Oliveira, A. Ferreira, M.A. Vilaça, H.C. Wong, M. Bern, R. Dahab, A.A.F. Loureiro, SecLEACH – on the security of clustered sensor networks, Signal Processing 87 (12) (2007) 2882–2895.

[27] R.J. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, P. Kruus, TinyPK: securing sensor networks with public key technology, in: S. Setia, V. Swarup (Eds.), Second ACM Workshop on Security of ad hoc and Sensor Networks (SASN'04), ACM Press, 2004, pp. 59–64.

[28] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and RSA on 8-bit CPUs, in: M. Joye, J.-J. Quisquater (Eds.), Workshop on Cryptographic Hardware and Embedded Systems (CHES'04), LNCS, Vol. 3156, Springer, 2004, pp. 119–132.

[29] D.J. Malan, M. Welsh, M.D. Smith, A public key infrastructure for key distribution in tinyos based on elliptic curve cryptography, in: First IEEE International Conference on Sensor and ad hoc Communications and Networks (SECON'04), 2004, pp. 71–80.

[30] W. Du, R. Wang, P. Ning, An efficient scheme for authenticating public keys in sensor networks, in: P.R. Kumar, A.T. Campbell, R. Wattenhofer (Eds.), Sixth ACM International Symposium on Mobile ad hoc Networking and Computing (MOBIHOC'05), ACM Press, 2005, pp. 58–67.

[31] K.G. Paterson, Cryptography from pairings, in: I.F. Blake, G. Seroussi, N. Smart (Eds.), Advances in Elliptic Curve Cryptography, London Mathematical Society Lecture Notes, Vol. 317, Cambridge University Press, 2005, pp. 215–251. Chapter X.

[32] C. Cocks, An identity-based encryption scheme based on quadratic residues, in: B. Honary (Ed.), Eighth IMA International Conference on Cryptography and Coding, LNCS, Vol. 2260, Springer, 2001, pp. 360–363.

[33] A. Joux, A one round protocol for tripartite Diffie–Hellman, Journal of Cryptology 17 (4) (2004) 263–276. also appeared in ANTS-IV.

[34] A. Menezes, T. Okamoto, S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, IEEE Transactions on Information Theory 39 (5) (1993) 1639–1646.

[35] S.D. Galbraith, Pairings, in: I.F. Blake, G. Seroussi, N. Smart (Eds.), Advances in Elliptic Curve Cryptography, London Mathematical Society Lecture Notes, Vol. 317, Cambridge University Press, 2005, pp. 183–213. Chapter IX.

[36] S.D. Galbraith, K.G. Paterson, N.P. Smart, Pairings for cryptographers, Discrete Applied Mathematics 156 (16) (2008) 3113–3121.

[37] P. Szczechowiak, A. Kargl, M. Scott, M. Collier, On the application of pairing based cryptography to wireless sensor networks, in: D.A. Basin, S. Capkun, W. Lee (Eds.), Second ACM Conference on Wireless Network Security (WISEC'09), ACM Press, 2009, pp. 1–12.

[38] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu, Analyzing and modeling encryption overhead for sensor network nodes, in: C.S. Raghavendra, K.M. Sivalingam, R. Govindan, P. Ramanathan (Eds.), Second ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03), ACM Press, 2003, pp. 151–159.

[39] P.S.L.M. Barreto, S. Galbraith, C. Ó hÉigeartaigh, M. Scott, Efficient pairing computation on supersingular abelian varieties, Designs Codes and Cryptography 42 (3) (2007) 239–271.

[40] M. Duursma, H.-S. Lee, Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$, in: C.-S. Laih (Ed.), Ninth International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'03), LNCS, Vol. 2894, Springer, 2003, pp. 111–123.

[41] Atmel, 8 bit AVR Microcontroller ATmega128(L) Manual, v. 2467M-AVR-11/04, November 2004.

[42] R.M. Avanzi, Another look at square-roots (and other less common operations) in fields of even characteristic, in: C.M. Adams, A. Mi ri, M.J. Wiener (Eds.), 14th International Workshop on Selected Areas in Cryptography (SAC 2007), LNCS, Vol. 4876, Springer, 2007, pp. 138–154.

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

[44] J. López, R. Dahab, High-speed software multiplication in GF(2^m), in: B.K. Roy, E. Okamoto (Eds.), First International Conference in Cryptology in India (INDOCRYPT'00), LNCS, Vol. 1977, Springer, 2000, pp. 203–212.

[45] K. Fong, D. Hankerson, J. López, A. Menezes, Field inversion and point halving revisited, IEEE Transactions on Computers 53 (8) (2004) 1047–1059.

[46] Atmel Corporation, AVR Studio 4, URL <http://www.atmel.com/>, 2005.

[47] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, T. Voigt, MSPsim – an extensible simulator for msp430-equipped sensor boards, in: Fourth European Conference on Wireless Sensor Networks (EWSN'07), Poster/Demo session, 2007, URL <http://www.sics.se/adam/eriksson07mspsim.pdf>.

[48] J. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, F. Rodríguez-Henríquez, A comparison between hardware accelerators for the modified tate pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$, in: S.D. Galbraith, K.G. Paterson (Eds.), Second International Conference on Pairing-Based Cryptography (Pairing'08), 2008, pp. 297–315.

[49] S. Bartolini, I. Branovic, R. Giorgi, 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 57 (5) (2008) 672–685.

[50] A. Liu, P. Kampanakis, P. Ning, TinyECC: elliptic curve cryptography for sensor networks (Ver. 0.3), URL <http://discovery.csc.ncsu.edu/software/TinyECC/>, 2005.

[51] C. Blundo, A.D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, Perfectly-secure key distribution for dynamic conferences, in: E.F. Brickell (Ed.), 12th Annual International Cryptology Conference (CRYPTO'92), LNCS, Vol. 740, Springer, 1992, pp. 148–167.

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013

[52] R. Blom, An optimal class of symmetric key generation systems, in: Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'84), 1984, pp. 335–338.
[53] Y. Zhang, W. Liu, W. Lou, Y. Fang, Securing sensor networks with location-based keys, in: IEEE Wireless Communications and Networking Conference (WCNC'05), 2005.
[54] B. Doyle, S. Bell, A.F. Smeaton, K. McCusker, N. O'Connor, Security considerations and key negotiation techniques for power constrained sensor networks, The Computer Journal 49 (4) (2006) 443–453.
[55] L.B. Oliveira, R. Dahab, J. Lopez, F. Daguano, A.A.F. Loureiro, Identity-based encryption for sensor networks, in: Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'07), 2007, pp. 290–294.
[56] L.B. Oliveira, D.F. Aranha, E. Morais, F. Daguano, J. López, R. Dahab, TinyTate: computing the tate pairing in resource-constrained nodes, in: Sixth IEEE International Symposium on Network Computing and Applications (NCA'07), 2007, pp. 318–323.
[57] L.B. Oliveira, M. Scott, J. López, R. Dahab, TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, in: Fifth International Conference on Networked Sensing Systems (INSS'08), 2008, pp. 173–180.
[58] M. Scott, MIRACL—A multiprecision integer and rational arithmetic C/C++ library, Shamus Software Ltd, Dublin, Ireland, URL <http://www.shamus.ie>, 2003.
[59] T. Ishiguro, M. Shirase, T. Takagi, Efficient implementation of pairings on sensor nodes, in: Applications of Pairing-Based Cryptography – NIST, 2008, pp. 96–106.
[60] P. Szczechowiak, L.B. Oliveira, M. Scott, M. Collier, R. Dahab, NanoECC: testing the limits of elliptic curve cryptography in sensor networks, in: Fifth European Conference on Wireless Sensor Networks (EWSN'08), 2008, pp. 305–320.
[61] L.B. Oliveira, A. Kansal, B. Priyantha, M. Goraczko, F. Zhao, Secure-TWS: authenticating node to multi-user communication in shared sensor networks, in: Eighth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'08), 2009, pp. 289–300.
[62] D. Boneh, B. Lynn, H. Schacham, Short signatures from the weil pairing, Journal of Cryptology 17 (4) (2004) 297–319.
[63] D. Galindo, R. Roman, J. Lopez, A killer application for pairings: authenticated key establishment in underwater wireless sensor networks, in: M.K. Franklin, L.C.K. Hui, D.S. Wong (Eds.), Seventh International Conference on Cryptology and Network Security (CANS'08), LNCS, Vol. 5339, Springer, 2008, pp. 120–132.

Please cite this article in press as: L.B. Oliveira et al., TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Comput. Commun. (2010), doi:10.1016/j.comcom.2010.05.013