Journal of Cryptographic Engineering manuscript No.
(will be inserted by the editor)

Efficient Software Implementation of Public-Key Cryptography on Sensor Networks Using the MSP430X Microcontroller

Conrado P. L. Gouvêa · Leonardo B. Oliveira · Julio López

Received: date / Accepted: date

Abstract In this work we describe a software implementation of Elliptic Curve Cryptography (ECC) and Pairing-Based Cryptography (PBC) for the MSP430 microcontroller family, which is used in wireless sensors. Digital signature, short signature and key distribution protocols were implemented at the 80- and 128-bit levels of security, over both binary and prime fields. The timing results of our software implementation show an improvement of about 25–30% in the pairing computation over previous implementations. We also provide results for the MSP430X extension of the original family, which has new instructions. In particular, using the new 32-bit hardware multiplier available in some MSP430X models, we have achieved a further improvement of about 45% in the prime field multiplication and 20–30% in protocol timings. The combination of fast algorithms and improved hardware allows us to show that even the 128-bit level of security can be considered feasible for this platform.

Keywords Efficient software implementation · Elliptic curve cryptography · Pairing-based cryptography · MSP430

1 Introduction

Wireless Sensor Networks (WSNs) are ad hoc networks consisted mainly of small sensor nodes with limited resources and one or more base stations, which connect the sensor nodes to the rest of the world. They are used for monitoring environments and provide fine-grained sensing to users.

Conrado P. L. Gouvêa · Julio López
University of Campinas, Campinas, Brazil
E-mail: conradoplg@ic.unicamp.br

Julio López
E-mail: jlopez@ic.unicamp.br

Leonardo B. Oliveira
Federal University of Minas Gerais, Belo Horizonte, Brazil
E-mail: leob@dcc.ufmg.br

Security in WSNs is of paramount importance because their applications range from battlefield reconnaissance and emergency rescue operations to surveillance and environmental protection. Besides, the fact that they monitor the environment raises innumerous privacy concerns that can be only mitigated through the use of security.

In this work, we describe a high speed software implementation of both Elliptic Curve Cryptography (ECC) and Pairing-Based Cryptography (PBC) for wireless sensors using the MSP430 family of microcontrollers (used in popular sensor nodes platforms such as the TelosB and the Tmote Sky). We show how one can combine algorithmic strategies and platform technology to efficient implement in software the Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic Curve Schnorr Signature (ECSS), and Zhang-Safavi-Naini-Susilo (ZSS) signature schemes as well as the Sakai-Ohgishi-Kasahara (SOK) and Elliptic Curve Menezes-Qu-Vanstone (ECMQV) key agreement protocols at both 80-bit and 128-bit security levels over both prime and binary fields.

Our main contributions are (i) to show how to implement these protocols efficiently on the MSP430; (ii) to our knowledge, present the fastest times published so far for the platform; (iii) assess the impact on performance of the MSP430X, the most recent representative of the MSP430 family; (iv) show how to take advantage of the new 32-bit hardware multiplier available in some MSP430X models; (v) and show the feasibility of the 128-bit level of security on modern MSP430 devices. Our results indicate improvements that ranges from 25% to 30% over previous timings.

The remainder of this work is organized as follows. In Section 2, we present the characteristics of the MSP430 family. In Section 3 we discuss the cryptographic protocols we have implemented. In Sections 4 and 5 we describe our implementation techniques and optimizations along with the results obtained. Finally, we discuss related work in Section 6 and conclude in Section 7.

Table 1: Features of relevant MSP devices

Microcontroller	MSPX	Clock (MHz)	ROM (KB)	RAM (KB)	MPY32
MSP430F1611	No	8	48	10	No
MSP430F2417	Yes	16	92	8	No
CC430F6137	Yes	20	32	4	Yes
MSP430F5529	Yes	25	128	8	Yes

2 The MSP430 Family

The microcontrollers from the MSP430 family have many characteristics in common, such as being 16-bit, having the same instruction set and 12 general-purpose registers. The clock frequency and ROM/RAM sizes varies for each member. Table 1 presents the features of some relevant microcontrollers, including the MSP430F1611 used in the Tmote Sky and TelosB sensors, and the MSP430F2417 used in the TinyNode 184.

The MSP430 family instruction set has addition, subtraction and one-bit only shifts. A swap byte instruction is available, which can be used for cheaper 8-bit shifts. Integer multiplication is carried out with a hardware multiplier — a memory-mapped peripheral that is present in all mentioned models. The cost of using this hardware is simply the cost of writing the operands and reading the result to/from a certain memory address. There is no division instruction. Operands can be referenced by four addressing modes: register direct, indexed, register indirect and indirect with auto-increment. Instructions can use immediate constants that are codified in offset words adjacent to the instruction. The number of cycles that an instruction takes to execute can be computed easily, with a few exceptions. It takes one cycle to fetch the instruction and one cycle to read each offset word, if any. Add one cycle for each in-memory source (read) and two cycles for a in-memory destination (write).

The MSP430 architecture was later expanded into the backward-compatible MSP430X architecture. These microcontrollers (also referred to as MSPX) are able to address up to 1 MB of memory with 20-bit pointers. New instructions are available, such as pushing and popping multiple registers with only one instruction and up to 4-bit shifts with one instruction (which still take the same number of cycles than using separate instructions). Its instructions timings are also different; the most important distinction is that moving data to memory takes one less cycle to execute. Some new MSP430X models feature a new 32-bit hardware multiplier (referred to as MPY32) whose usage is similar to the old 16-bit multiplier and which can be used to greatly improve the performance of cryptographic operations as will be described later.

Table 2: Elliptic curves used in this work

Curve	Security (bits)	Field	Random	Used in
secp160r1 [10]	80	Prime	Yes	ECC
secp160k1 [10]	80	Prime	No	ECC
K-163 [34]	80	Binary	No	ECC
BN-158 [37]	80	Prime	No	PBC
SS-353 [37]	80	Binary	No	PBC
P-256 [34]	128	Prime	Yes	ECC
secp256k1 [10]	128	Prime	No	ECC
K-283 [34]	128	Binary	No	ECC
BN-254 [35]	128	Prime	No	PBC

3 Cryptographic Protocols

An elliptic curve E over a field K is defined by the equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, with $a_i \in K$. A point in an elliptic curve is a pair $(x,y) \in K \times K$ that satisfies the curve equation. It is possible to define a point addition operation such that the set of points in the curve — together with an identity element called the point at infinity — forms a group. This group is denoted $E(K)$. Given a point $P = (x_P, y_P)$ in $E(K)$ and an integer k, the point multiplication operation kP can be defined as the sum of k copies of the point P, using the point addition operation of the elliptic curve. Table 2 lists the curves used in this work. A random curve is a curve which was generated verifiably at random.

Given three groups $\mathbb{G}_1$, $\mathbb{G}_2$ e $\mathbb{G}_T$ of prime order r, with $\mathbb{G}_1$ e $\mathbb{G}_2$ additive and $\mathbb{G}_T$ multiplicative, a bilinear pairing is defined as a map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the bilinearity property, that is, for all $a, b \in \mathbb{Z}$, $R \in \mathbb{G}_1$ and $S \in \mathbb{G}_2$ we have that $e(aR, bS) = e(R,S)^{ab}$. In PBC, usually, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of the group of points in certain elliptic curves over finite fields and $\mathbb{G}_T$ is the multiplicative subgroup of a finite field. If $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is called symmetric, otherwise, it is called asymmetric. In this work, the pairing over the binary field is symmetric and the pairing over the prime field is asymmetric.

3.1 ECDSA

We have implemented the Elliptic Curve Digital Signature Algorithm (ECDSA) [34] since it is extremely popular and standardized. It is composed of three operations: key generation, signature and verification. The key generation requires a point multiplication of a fixed point G that is known by all participants. The signature also requires a fixed point multiplication while the signature requires the computation of $kG + \ell Q$ where G is fixed and Q is the point corresponding to the public key of the signer. For more details we refer to the standard [34].

For implementing ECDSA we have chosen the curves secp160r1 and secp160k1 (over 160-bit prime fields) from

the SECG standard [10]; P-256 and secp256k1 (over 256-bit prime fields) from the NIST and SECG standards [10,34]; K-163 (over the binary field $\mathbb{F}_{2^{163}}$) and K-283 (over $\mathbb{F}_{2^{283}}$) from NIST [34]. Those curves allow a fast modular reduction with only additions, shifts and xors (and a few multiplications in some cases) due to the special form of their prime moduli and reduction polynomials.

3.2 ECSS

We have also implemented the Elliptic Curve Schnorr Signature (ECSS) [39] which is very similar to ECDSA but does not require an inversion modulo the group order, thus being faster. The ECSS has been patented, but its patents seem to have expired recently. The same curves used in ECDSA were employed for ECSS.

3.3 ZSS Short Signature Scheme

The Zhang-Safavi-Naini-Susilo (ZSS) scheme [47] is a short signature scheme based on pairings that allows signatures with half the size of ECDSA's. Its signature generation requires a fixed point multiplication, while the verification requires one pairing computation and one fixed point multiplication.

We remark that using supersingular binary curves in order to implement pairing-based short signatures defeats the short signature property, as was pointed out in [9]. Since these curves have a small embedding degree, they require larger fields and therefore provide larger signatures. For example, at the 80-bit level of security, a 353-bit field is required which results in a 353-bit signature, in contrast to the 320-bit signature achieved by ECDSA. For this reason, only prime fields were used for this protocol. Two BN curves [6] over prime fields with 158 bits (BN-158) and 254 bits (BN-254) were chosen, using the Optimal Ate pairing [45].

3.4 ECMQV Authenticated Key Agreement Protocol

The Elliptic Curve Menezes-Qu-Vanstone (ECMQV) [27] is a family of authenticated key agreement protocols; we have implemented the two-pass variant. This protocol is composed of two operations: key generation and key agreement. The key generation is the same as ECDSA's and requires a fixed point multiplication. The key agreement itself has two stages: first, the parties exchange public keys; then, a ephemeral key pair is generated (again with a fixed point multiplication) and the ephemeral public key is sent to the other party, while the ephemeral public key of the other party is received. The agreed key is then computed using two random point multiplications. The curves used for ECMQV were the same curves used for ECDSA.

3.5 SOK Non-Interactive Authenticated Key Agreement Protocol

Public key authentication is the main problem of ECC, since it requires an infrastructure that can be expensive for sensors. The Identity-Based Cryptography (IBC) provides an alternative to classic asymmetric cryptography where public keys are the identities of the users (such as an email address) and thus they are implicitly authenticated. Its downside is the requirement for a Key Generation Center (KGC) which generates the private keys of the users and therefore can impersonate them. However, this center is acceptable in the wireless sensor network scenario since the sensors trust the KGC as they are all maintained by the same organization. The most popular method to instantiate IBC is based on pairings.

One of the best IBC schemes for wireless sensors networks is the Sakai-Ohgishi-Kasahara (SOK) non-interactive authenticated key agreement protocol [38]. This protocol enables two parties to combine a mutual key without any communication, knowing only each other's identities; afterwards, communication can be carried out with symmetric encryption. For this reason, it is well suited for sensor networks where communication is often costly. In SOK, a key agreement requires a single pairing computation. If the pairing is asymmetric, its implementation is slightly more complex. While [14] suggests computing two pairings to solve the asymmetric case, it can be done with only one pairing computation [19].

Loosely speaking, the security of SOK depends on the difficulty of solving the discrete logarithm problem in the pairing groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Since $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curves, the underlying fields must have at least 160 and 256 bits at the 80- and 128-bit levels of security, respectively. The finite field from which $\mathbb{G}_T$ is a subgroup, however, must have a larger size than $\mathbb{G}_1$ and $\mathbb{G}_2$ due to the existence of sub-exponential attacks to its discrete logarithm problem. For prime fields, the standard [33] recommends orders with 1024 and 3072 bits at the 80- and 128-bit levels of security, respectively. For binary fields, due to Coppersmith's attack [12], a more conservative approach is required: at least approximately 1412 and 4036 bits at the 80- and 128-bit levels, respectively.

In order to implement SOK the two aforementioned BN curves were used (BN-158 and BN-254) along with a supersingular curve over the binary field $\mathbb{F}_{2^{353}}$ (SS-353) [37]. Note that the curve over $\mathbb{F}_{2^{271}}$ used in works such as [36,43] offers only 70 bits of security due to [12], while the curve over $\mathbb{F}_{2^{353}}$ offers 80 bits of security. A binary curve offering 128 bits of security was not implemented since it would required a huge base field (more than 1000 bits) and thus would result in very poor performance.

4 Efficient Software Implementation

Our implementation was written in the C language, with critical finite field operations written in assembly. The IAR Embedded Workbench 5.40 was used for development and execution, using both its simulator and boards equipped with a MSP430F2417 and a CC430F6137.

4.1 Prime Field Arithmetic

The prime field arithmetic is composed of addition, subtraction, multiplication, squaring and inversion modulo a prime p. The most important are multiplication and squaring, since ECC and PBC protocols spend around 70% of their computation in those operations. The multiplication in a prime field has two steps: the integer multiplication of the two n-word operands into a $2n$-word intermediate result, and the reduction modulo p in order to obtain the n-word result.

The integer multiplication is carried out with the Comba algorithm [11], a column-oriented version of the usual multiplication algorithm. There is also a hybrid algorithm that combines both techniques [41]. However, Comba offers a better performance in this platform since its fundamental multiply-and-accumulate step is exactly what is provided by the Multiply and Accumulate (MAC) operation of the MSP430 hardware multiplier [19]. The Karatsuba [25] algorithm can also be used in order to reduce the size of the multiplier required. We have determined that using Karatsuba leads to a better performance in the 256-bit field, and thus was used in this case.

The modular reduction of the intermediate result is computed with the Montgomery algorithm [32], since it does not requires division, which is not available in the MSP430. It has the same structure as the Comba multiplication, but with the first operand being the prime modulus and the second operand generated on the fly during the algorithm. Therefore, we can use the same MAC optimization [19].

When using the 32-bit multiplier, we employ the same Comba method, now with 32-bit digits and still taking advantage of the MAC operation. In order to fully use the power of the multiplier, it is important to enable the MPYDLYWRTEN bit which allows writing the operands of the next MAC operation while the previous is still being carried out.

In the PBC context, a very effective optimization for modular reduction was found, which was briefly mentioned in [37] and is further detailed here. When using BN curves, the prime modulus is given by the polynomial $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ where x is the parameter that defines the curve. Consider the curves generated by the values $x_1 = 2^{38} + 2^5 + 2^4 + 1$ (suitable at the 80-bit level of security) and $x_2 = -2^{62} - 2^{55} - 1$ (suggested by [35], suitable at the 128-bit level of security). The prime moduli in those cases are

$$p(x_1) = \texttt{0x2400 0000 6ED0 0000 7FE9}$$
$$\texttt{C000 419F EC80 0CA0 35C7},$$
$$p(x_2) = \texttt{0x2523 6482 4000 0001 BA34 4D80 0000 0008}$$
$$\texttt{6121 0000 0000 0013 A700 0000 0000 0013},$$

respectively, in base 16. Notice that $p(x_1)$ has two 16-bit digits with a zero value, while $p(x_2)$ has five digits with zero value and one digit is the number one. Since Montgomery reduction is analogous to a multiplication, the steps with multiplications by zero can be discarded and the steps with multiplication by one can be simplified. For example, when using $p(x_2)$, it was required $16^2 = 256$ multiply-and-accumulate steps during the Montgomery reduction. Ignoring the steps with multiplications by zero, $256 - 16 \cdot 5 = 176$ steps are now required, a 31% reduction. We remark that the generic optimization technique that takes advantage of sparse primes has been used before [21,26], but to the best of our knowledge this work and [37] provide the first application of the technique in the context of PBC. The focus of the curve suggested by [35] was probably in the fact that a sparse parameter leads to a faster Miller loop, but it also speeds up the field arithmetic as shown.

When using the 32-bit multiplier, it is more difficult to apply this technique. In $p(x_2)$, the three zeros in odd-indexed 16-bit digits allow the replacement of some 32x32-bit multiplications by 32x16-bit multiplications, which is a little faster. However, we were unable to take advantage of the zeros in even-indexed digits.

4.2 Binary Field Arithmetic

The binary field arithmetic is composed by the addition, subtraction, multiplication, squaring, inversion and square root operations over the field $\mathbb{F}_{2^m}$. Addition in this field becomes a simple xor of the operands; multiplication becomes more expensive due to the absence of a hardware multiplier and therefore must be implemented very efficiently; and squaring can be computed much faster with precomputed tables.

The binary field multiplication is also composed of two steps: polynomial multiplication of the operands followed by a polynomial reduction. The polynomial multiplication is computed with the López-Dahab (LD) algorithm [29], which appears to be the most efficient in this scenario [3, 43]. Optionally, the LD algorithm can be combined with the Karatsuba algorithm; we have employed it in the 283- and 353-bit fields. The square of a binary field element can be computed very quickly with the help of precomputed tables stored in ROM, as described in e.g. [3].

The polynomial reduction after polynomial multiplication and squaring can be computed efficiently due to the special form of the reduction polynomial in the chosen curves.

It can be computed with only shifts and xors, as described in e.g. [23]. For the curve over $\mathbb{F}_{2^{353}}$ used in the η_T pairing, we have used the trinomial $z^{353}+z^{95}+1$ for the field $\mathbb{F}_{2^{353}}$ since reduction, in this case, can be carried out with 1- and 2-bit shifts only [37].

The square root of a binary field element $a(z)$ is the element $c(z)$ such that $c(z)^2 = a(z)$. In this work, it is only used on the curve $\mathbb{F}_{2^{353}}$, during the computation of the η_T pairing. As described in [15], it can be computed using lookup tables and a multiplication by $\sqrt{z}$. This multiplication can be computed efficiently in the given curve, using 1-bit shifts only, by noting that $\sqrt{z} \equiv z^{177}+z^{48}$ in $\mathbb{F}_{2^{353}}$.

Inversion in the binary field can be computed using the extended Euclidean algorithm. In order to optimize it we have used the same approach as [3], using dedicated functions for shifting one to eight bits which are timing consuming during inversion.

4.3 Point Arithmetic

The main operations in ECC are the the random point multiplication (kP), fixed point multiplication (kG, where G is known in advance) and the simultaneous multiplication of a fixed and a random point ($kG+\ell P$), all of which are based on point addition and duplication. Using the names from the Explicit-Formulas Database[1], the prime curves use Jacobian coordinates and the `madd-2007-bl` mixed addition [8] formulas; the `dbl-2001-b` doubling [7] formulas were used for the secp160r1 and P-256 curves while the `dbl-2009-l` doubling [4] formulas were used for the secp160k1, secp256k1, BN-158 and BN-254 curves (except for the pairing computation in the latter two curves). For the K-163 and K-283 curves, the `madd-2005-dl` mixed addition formula [1] was employed; point doubling is not needed for Koblitz curves.

For fixed point multiplication, the Comb [28] method was selected in the prime case and the wTNAF method in the binary case [42]. These methods take advantage of precomputation tables containing multiples of the fixed point G in order to speed up the computation. For simultaneous multiplication, the wNAF- and wTNAF-based interleaving method [31] were chosen for the prime and binary cases, respectively. This method uses a precomputation table for the fixed point and a smaller table, computed on the fly, for the random point. For all methods mentioned, we have used precomputed tables with 8 points and on-the-fly tables with 4 points.

On the secp160k1 and secp256k1 curves, which are used for ECDSA and ECSS, and on the BN curves used for ZSS, the Gallant-Lambert-Vanstone (GLV) method [18] enables a faster point multiplication. These curves all have the form $y^2 = x^3+b$ over $\mathbb{F}_p$ with $p \equiv 1 \pmod 6$ and have a map $\phi\colon E(\mathbb{F}_p) \mapsto E(\mathbb{F}_p)$ defined by $(x,y) \mapsto (\beta x, y)$ where $\beta \in \mathbb{F}_p$ is an element of order 3. Let λ be an integer satisfying $\lambda^2+\lambda \equiv -1 \pmod n$. Then $\phi(P) = \lambda P$. It is possible to take advantage of this fact in order to improve the speed of point multiplication kP, as follows. Write k as $k = k_0 + \lambda k_1$, then compute $k_0 P + k_1 \phi(P)$. If k_0 and k_1 have half the bit length of k, then it is possible to use the interleaved point multiplication [31] which is faster than computing kP directly, since the number of point duplications will be cut in half.

The critical operation in the signing algorithm of ZSS, ECDSA, and ECSS is the fixed point multiplication, where the Comb method is often faster than wNAF. However, mixing Comb and GLV is not straightforward. It can be done, as mentioned in [37], by considering the two-table version of the Comb method. In this version, two tables are precomputed with multiples of P and $2^{t/2}P$, where t is the bit length of k. Write $k = k_0 + 2^{t/2}k_1$; now, kP can be computed taking advantage of the two tables. When the GLV method is available, we can decompose k as usual ($k = k_0 + \lambda k_1$). However, the second table, now with multiples of $\phi(P)$, does not need to be precomputed since we can just apply ϕ to the elements of the first table when necessary. Therefore, we can use half of the space of the usual Comb multiplication while achieving the same performance, or we can use the same space (doubling the size of the first table) while cutting the number of point duplications in half — which we have chosen.

For GLV to be effective, the decomposition of k must be computed efficiently. This can be done using some precomputation, as hinted in [23] and mentioned in [37]. We describe the approach in [37] in a more detailed manner as follows. In order to decompose k as k_0 and k_1 with the appropriate bit lengths, we precompute the vectors $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{Z}\times\mathbb{Z}$ as described in [18]. Now, we must solve the equation $(k,0) = \beta_1\mathbf{v_1} + \beta_2\mathbf{v_2}$ for β_1 and β_2, then round the result to the nearest integers as $b_1 = \lfloor \beta_1 \rceil$ and $b_2 = \lfloor \beta_2 \rceil$. Compute $\mathbf{v} = b_1\mathbf{v_1} + b_2\mathbf{v_2}$ and finally let $(k_0,k_1) = (k,0) - \mathbf{v}$.

The critical operation when decomposing k is to solve the given equation. Write $\mathbf{v_1} = (v_{10}, v_{11})$ and $\mathbf{v_2} = (v_{20}, v_{21})$. Let $d = v_{10}v_{21} - v_{11}v_{20}$. It is then possible to compute β_1, β_2 as $\beta_1 = (kv_{21})/d$, $\beta_2 = -(kv_{11})/d$. The multiple precision division by d is the most expensive part, but it can be avoided as follows. Let $t = \lfloor k \rfloor + 2$ and compute $g_1 = \lfloor (2^t v_{21})/d \rceil$ and $g_2 = \lfloor (2^t v_{11})/d \rceil$. Note that g_1 and g_2 are integer constants which can be precomputed. We can now compute both $\beta_1' = \lfloor (kg_1)/2^t \rfloor$, $\beta_2' = -\lfloor (kg_2)/2^t \rfloor$, where the floored division by 2^t is a simple right shift of t bits. The last bit discarded in this right shift must be stored, and if it is 1, then $b_1 = \beta_1' + 1$, otherwise $b_1 = \beta_1$. The same applies to β_2'. The decomposition of k now follows as described.

[1] `http://www.hyperelliptic.org/EFD/`

4.4 Pairing Computation Algorithms

The standard algorithm for computing pairings is the Miller algorithm [30]. The first pairings used were the Weil and Tate, but the fastest ones are called optimal pairings [45]. Among them, the Optimal Ate pairing was chosen [45] for the prime case and the η_T pairing [5] for the binary case. A pairing computation is divided in two parts: the Miller loop, which is analogous to a point multiplication with some additional computation; and the final exponentiation, which maps the result of the Miller loop — a coset element — to a canonical representation of the coset.

In this work, BN curves [6] were used in the prime case. They are a family of pairing-friendly elliptic curves of the form $y^3 = x^3 + b$, with embedding degree 12. When using a BN curve with the Optimal Ate pairing, the pairing groups become $\mathbb{G}_1 = E(\mathbb{F}_p)$, $\mathbb{G}_2 = \{Q \in E(\mathbb{F}_{p^{12}}) : pQ = (x_Q^p, y_Q^p), rQ = \infty\}$ and $\mathbb{G}_T = \{x \in \mathbb{F}^*_{p^{12}} : x^r = 1\}$. The $\mathbb{G}_2$ group can be represented by the preimage of the twisting isomorphism $\phi : E' \to E$, where E' is the sextic twist of E. Since this group is defined over $\mathbb{F}_{p^2}$, it has smaller elements and its operations are more efficient. The Miller loop therefore requires point arithmetic in $E'(\mathbb{F}_{p^2})$. Points are represented with projective coordinates using the formulas from [2]. The loop also requires arithmetic in $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^{12}}$. The latter is built as an extension tower $\mathbb{F}_{p^2} \to \mathbb{F}_{p^6} \to \mathbb{F}_{p^{12}}$. In $\mathbb{F}_{p^2}$ we employ the lazy reduction technique [46]. This allows computing $ab + cd$, for example, with two multiplications and a single reduction, instead of the usual two multiplications and two reductions — this is accomplished by multiplying ab and cd, adding the double precision results, and reducing the result. It is possible to employ this technique for the entire $\mathbb{F}_{p^{12}}$ as described in [2], but it can be costly in terms of memory, as will be examined later.

Still in the prime case using BN curves, the final exponentiation requires $\mathbb{F}_{p^{12}}$ arithmetic. We follow the approach from [40], with the field squaring optimized as described in [20]. It is also possible to employ the compressed squaring technique from [24] as detailed and improved in [2] but again it can be costly in terms of memory. Hashing to $\mathbb{G}_2$ in the asymmetric case is another important operation since it requires a expensive point multiplication by the cofactor of $E'(\mathbb{F}_{p^2})$, which has the same size of p. However, this multiplication can be greatly sped up using the technique from [16].

For the binary case, a supersingular curve over a 353-bit field was used. Here, the pairing groups become $\mathbb{G}_1 = \mathbb{G}_2 = E(\mathbb{F}_{2^{353}})$ and $G_T = F_{2^{4\times 353}}$. The η_T pairing computation follows its original description [5].

5 Results

Using the MSPsim and the IAR simulators, the number of cycles taken by each operation was measured. In some cases we give the timing in seconds; these are derived from the number of cycles assuming a 8 million hertz clock. The exact clock depends on the MSP430 model being used and in the voltage fed to the microcontroller.

The MSP430F6137 microcontroller used for testing the 32-bit multiplier features only 4 KB of RAM and 32 KB of ROM, making it difficult to fit our code in the available space. In this case, we have used the IAR simulator to run the programs for the MSP430F5529, which has 128 KB of ROM and the same 32-bit multiplier.

5.1 Prime Field Timings

Table 3 gives the timings of the prime field operations. First, we note that there is a small gain in comparison to [19], which presents the fastest timings so far for the MSP430 to the best of our knowledge. Particularly, in the modular reduction, we have obtained an 8% improvement by noting that is not necessary to read the prime modulus from the memory — since it is fixed, we can embed its digits in the instructions as immediate constants. However, the most expressive savings come from the sparse modulus optimization — 14% in the 160-bit field and 26% in the 256-bit field.

The MSP430X improves the timings by roughly 15%, while the MPY32 results in multiplications and squarings that are around 40% faster and reductions that are around 30% faster. The limited improvement for reduction is due to not being able to take full advantage of the sparse primes.

5.2 Binary Field Timings

Timings for binary field arithmetic are listed in Table 4. Previous works have used the 70-bit level for PBC [36,43], which prevents comparisons. The reduction in $\mathbb{F}_{2^{283}}$ requires many shifts due to standardized reduction polynomial used in the K-283 curve; this explains why it is slower than the reduction in the larger field $\mathbb{F}_{2^{353}}$. The MSP430X improves the timings around 5–15%, less than in the prime case probably due to the smaller number of memory writes in the LD multiplication.

5.3 Protocol Timings

Table 5 presents the timings of signature algorithms. The fastest timings are plotted for comparison in Fig. 1 and 2. For ECDSA, it can be seen that the binary curves offer a better performance than the random prime curves, but the

Table 3: Timings in cycles for prime field arithmetic

	80-bit			128-bit		
Algorithm	MSP	MSPX	MPY32	MSP	MSPX	MPY32
Comba mult.	1 565	1 299	741	3 563	2 981	1 620
Comba squaring	1 350	1 056	630	2 946	2 435	1 369
Reduction						
Montgomery, in [19]	1 785			3 989		
Montgomery	1 659			3 600		
Montgomery, sparse	1 413	1 174	853	2 670	2 232	1 695

Table 4: Timings in cycles for binary field arithmetic

	80-bit, ECC		128-bit, ECC		80-bit, PBC	
Algorithm	MSP	MSPX	MSP	MSPX	MSP	MSPX
LD mult.	3 907	3 585	8 655	8 166	13 868	12 902
Squaring	249	199	389	325	489	415
Square root					934	852
Fast reduction	448	397	888	835	494	426

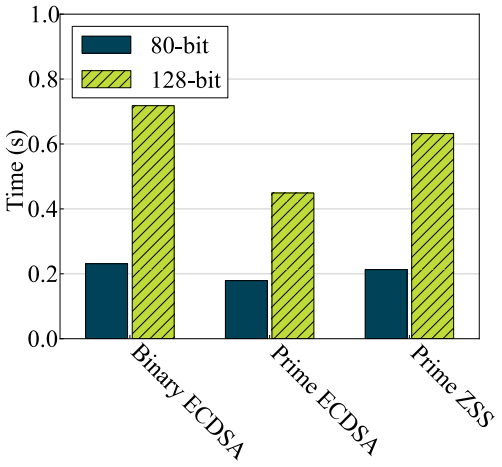

Fig. 1: ECDSA and ZSS signature timings for MSP430X with MPY32 at 8 MHz

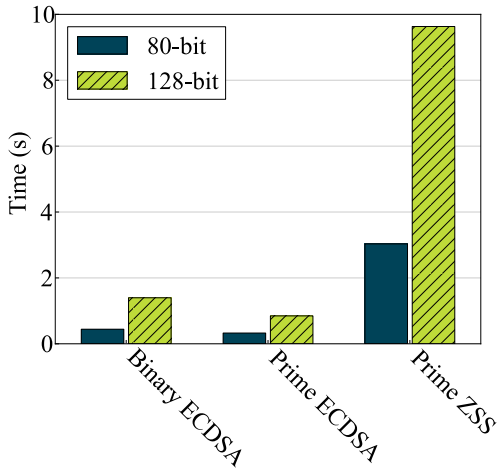

Fig. 2: ECDSA and ZSS verification timings for MSP430X with MPY32 at 8 MHz

special prime curves are even faster. Using MSP430X leads to a approximately 15% improvement in the prime case and around 5–10% in the binary case. Using MPY32 gives a further 20–30% improvement. Timings for ECSS were omitted for brevity but they are roughly 10% faster than ECDSA for signing and 5% faster for verifying.

Even though ZSS provides signatures with half the bit length of ECDSA, it appears that it does not give any substantial advantage in this scenario — the energy cost of the increased signing time defeats any energy saving with the decreased communication cost due to the smaller signature; as already shown in [37]. Nevertheless, it can be useful in scenarios where communication is very expensive, such as underwater sensor networks [17]. Also, note that verification in ZSS is costly since it requires one pairing computation; but this may not be a issue if only signing is required by the sensors and verification is done by more powerful devices.

Timings for pairing computation and key agreement protocols are given in Table 6 and the fastest version of each plotted for comparison in Fig. 3. We observe again a near 15% improvement with MSP430X and a further 25–30% gain with MPY32. Using MPY32, the prime curve almost reaches the performance of the binary curve at the 80-bit level of security. The reason for the binary curve being so

Table 5: Timings in seconds for signature protocols

		Sign			Verify		
Protocol	Curve	MSP	MSPX	MPY32	MSP	MSPX	MPY32
80-bit							
ECDSA	secp160r1	0.315	0.266	0.218	0.625	0.549	0.449
	secp160k1	0.254	0.214	0.179	0.450	0.387	0.327
	K-163	0.256	0.231		0.481	0.443	
ZSS	BN	0.356	0.284	0.213	5.908	4.077	3.032
128-bit							
ECDSA	P-256	0.924	0.807	0.557	1.882	1.671	1.156
	secp256k1	0.719	0.618	0.449	1.275	1.099	0.847
	K-283	0.772	0.718		1.474	1.397	
ZSS	BN	1.056	0.890	0.632	16.611	14.027	9.631

Table 6: Timings in seconds for pairing computation and key agreement protocols

Algorithm	Curve	MSP	MSPX	MPY32
80-bit				
η_T		2.586	2.395	
Optimal Ate		3.791	3.202	2.470
SOK	SS-353	2.668	2.493	
SOK ($\mathbb{G}_1$)	BN-158	3.875	3.285	2.533
SOK ($\mathbb{G}_2$)	BN-158	4.554	3.861	2.982
ECMQV	K-163	0.719	0.658	
	secp160r1	0.982	0.838	0.671
	secp160k1	0.730	0.626	0.513
128-bit				
Optimal Ate		9.930	8.461	5.967
SOK ($\mathbb{G}_1$)	BN-254	10.202	8.693	6.134
SOK ($\mathbb{G}_2$)	BN-254	11.766	10.036	7.121
ECMQV	K-283	2.291	2.150	
	secp256r1	2.842	2.437	1.742
	secp256k1	2.065	1.776	1.298

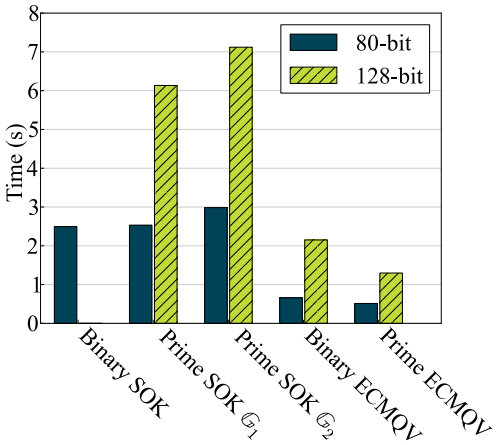

Fig. 3: SOK and ECMQV key agreement timings for MSP430X with MPY32 at 8 MHz

competitive — despite the fact that there is no hardware accelerator involved in its computation — is that, in both Koblitz curves and in the η_T pairing, point doublings are replaced by the Frobenius endomorphism which is simply the extremely fast squaring of the point coordinates. The ECMQV protocol is around 75–80% faster than SOK, but of course it requires a great deal of communication (to receive the certificate with the other party's public key, to send our certificate to the other party, to receive the other party's ephemeral public key, and to send our ephemeral public key to the other party) which makes it much less attractive in this scenario. It is interesting to note that the SOK with hash in $\mathbb{G}_2$ is more expensive than in $\mathbb{G}_1$ in the asymmetric case due to the requirement of multiplying the hashed point by the curve cofactor.

All timings reported so far have been scaled to a 8 MHz clock in order to allow comparisons with previous works. However, there are MSP430X microcontrollers with up to 25 MHz maximum clock. For this reason, to give a better picture of what is achievable with current technology, we report the timings using the 25 MHz clock in Table 7 for the best choices of the implemented protocols: ECDSA, ZSS, and ECMQV over GLV-enabled prime curves; SOK over binary curve at the 80-bit level and over BN curve at the 128-bit level. We remark that it is possible to sign in 0.2 s at the 128-bit level of security, and to compute a SOK key agreement in less than one second at the 80-bit level and under 2.5 s at the 128-bit level.

5.4 ROM, RAM and Energy

Figure 4 shows the ROM usage of the programs, while Fig. 5 shows the global RAM and the maximal stack RAM usage (ECSS and ECMQV are roughly the same as ECDSA). The ROM usage falls into the 23–42 KB range, with ECC protocols taking less space. The RAM usage falls into the 2–8 KB range. The ROM usage, while high, may be acceptable in

Table 7: Timings in seconds for best schemes at 25 MHz

	80-bit		128-bit	
Algorithm	MSPX	MPY32	MSPX	MPY32
ECDSA sign	0.068	0.057	0.198	0.144
ECDSA ver.	0.124	0.105	0.352	0.271
ZSS sign	0.091	0.068	0.285	0.202
ZSS ver.	1.305	0.970	4.489	3.082
ECMQV	0.200	0.164	0.568	0.415
SOK ($\mathbb{G}_1$)	0.798	0.798	2.782	1.963
SOK ($\mathbb{G}_2$)	0.798	0.798	3.212	2.279

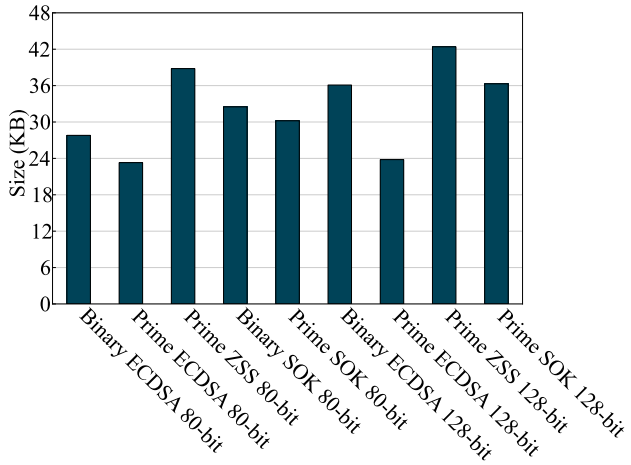

Fig. 4: ROM usage for ECC and PBC protocols

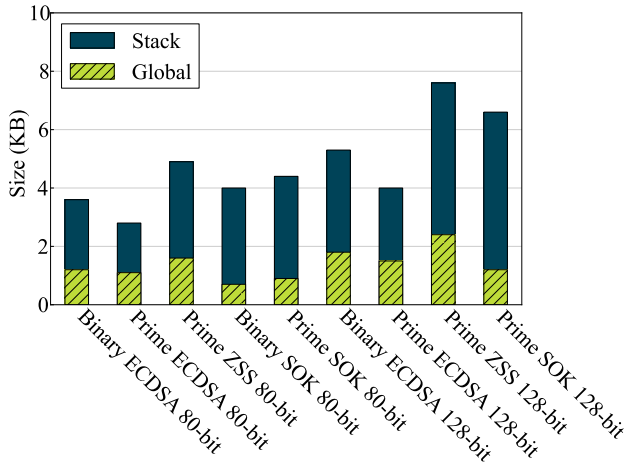

Fig. 5: RAM usage for ECC and PBC protocols

devices such as the MSP430F5529 that have a 128 KB flash. The RAM usage can also be considered acceptable by noting that most part of it is allocated from the stack and freed after computation. More RAM may be freed by moving the precomputed tables to ROM as a trade off.

Programs using prime fields are smaller due to the hardware multiplier which allows computation with a smaller number of instructions. The MSP430X and MPY32 also reduce the ROM usage by up to 800 bytes and 1.6 KB, respectively. Energy consumption seems to be not critical in this scenario by considering that with a 500 mAh battery it is possible to compute the most expensive operation (SOK key agreement in $\mathbb{G}_2$) roughly about 40 000 times, assuming a 3.31 mA current under 3.6 V, which are required to obtain the desired clock [13].

5.5 Impact of recent optimizations

Using the optimizations from [2] it is possible to speed up the prime pairing computation even more, but with a somewhat large memory cost. At the 80-bit level of security, using the lazy reduction for the entire extension tower leads to a 7–8% speed up in the pairing computation, but increases RAM usage by 1 KB (a 33% increase) and ROM usage by 0.9 KB. Using the compressed squaring optimization for the final exponentiation leads to a further 4% speed up, but increases RAM usage by another 1 KB and ROM usage by 1.4 KB. At the 128-bit level of security, the lazy reduction optimization leads to a 5% speed up (which is smaller than the 80-bit level since the reduction here is comparatively faster due to the sparser prime), increasing RAM usage by 1.5 KB (a 33% increase) and ROM usage by 1.2 KB; while the compressed squaring leads to a further 7% speed up, increasing RAM usage by 1.2 KB and ROM usage by 1.4 KB.

6 Related Work

One of the first ECC implementations for the MSP430 is the work of Guajardo et al. [22], which achieves a random point multiplication in 0.425 s (when scaled to 8 MHz) on the curve secp128r1 (64 bits of security), using the binary point multiplication algorithm and the 16-bit hardware multiplier. For comparison, our timing for the same algorithm is 0.581 s on the curve secp160r1, which offers 80 bits of security.

The work NanoECC [44] presents a point multiplication in 0.72 s and 1.04 s on prime (secp160r1) and binary (K-163) curves, respectively, at the 80-bit level of security. Unfortunately it is not clear which point multiplication algorithm was used in these timings. For comparison, our timings are 0.525 s and 0.256 s, respectively, using 4NAF/4TNAF. They also provide timings for the 7.4 MHz ATmega: 1.27 s and 2.16 s respectively.

In [19], it is described how to use the multiply and accumulate (MAC) operation of the MSP430 hardware multiplier to improve the speed of both ECC and PBC when

using prime curves. At the 80-bit level of security, our pairing over the BN curve offers a 25% gain in comparison to their pairing over a MNT curve (which offers roughly 70 bits of security). This result may seem surprising since the BN curve uses a $\mathbb{G}_T$ with 1920-bit order, much more than the 1024 bits required at this level of security. However, it is explained by the many existent optimizations tailored for BN curves. At the 128-bit level of security, we have obtained a 30% improvement in comparison their work in the pairing computation over the BN curve. This saving is explained by the sparse prime reduction optimization and the new formula for the final exponentiation from [20].

TinyPBC [36] provides timings for the η_T pairing over a supersingular 271-bit curve, which offers roughly 70 bits of security. The pairing timing is 1.27 s for the 8 MHz MSP430, 1.90 s for the 7.4 MHz ATmega and 0.14 s for the 13 MHz ARM PXA27x. For comparison, our 80-bit security pairing timing is 2.586 s. It is possible to notice that the binary pairing scales very poorly with the level of security.

The work Secure-TWS [37] focuses on digital signatures at the 80-bit level of security and presents a small subset of the results of this paper. The 7.4 MHz ATmega timings are 0.710 s for ZSS signature over prime BN curve, 0.680 s for ECDSA signature over the secp160k1 curve and 0.370 s over the K-163 curve.

7 Conclusion

Even though it is challenging, the implementation of cryptography for wireless sensor networks is viable. In this work, the best known timings for ECC and PBC in the MSP430 family of microcontrollers were presented, including results for the MSP430X extension of the family and using the new 32-bit hardware multiplier featured in some MSP430X models. Specifically, we have obtained a prime field multiplication that is 12% and 18% faster for 160- and 256-bit prime fields, respectively, by taking advantage of the sparse prime reduction. Our efficient implementation leads to a 25–30% speedup in the pairing computation compared to the best known timings published. We note that the sparse prime reduction can also be applied in other 8- and 16-bit platforms, and also at larger levels of security. Additionally, we were able to improve the timings of ECDSA and ZSS signatures using the GLV method, describing a method for decomposing the multiplier that avoids divisions and a method for using GLV in conjunction with the Comb point multiplication.

It was shown that the MSP430X extension provides a performance gain of roughly 15% due to faster instructions (mainly when writing to memory) and we were able to take advantage of the 32-bit multiplier present in some of the MSP430X microcontrollers in order to obtain a 20–30% improvement in protocol timings. Finally, we remark that on a 25 MHz MSP430F5529 with full clock speed it is possible to compute a 80-bit security SOK key agreement in 0.8 s, a 128-bit SOK key agreement in less than 2.5 s, and an ECDSA signature in under 150 ms.

Acknowledgements We would like to thank the São Paulo Research Foundation (FAPESP), which supported author Conrado Gouvêa under grant 2010/15340-3. We also thank Diego Aranha for the help with the implementation.

References

1. Al-Daoud, E., Mahmod, R., Rushdan, M., Kilicman, A.: A new addition formula for elliptic curves over GF(2^n). IEEE Transactions on Computers **51**(8), 972–975 (2002). DOI 10.1109/TC.2002.1024743
2. Aranha, D., Karabina, K., Longa, P., Gebotys, C., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Advances in Cryptology — EUROCRYPT 2011, *Lecture Notes in Computer Science*, vol. 6632, pp. 48–68. Springer Berlin / Heidelberg (2011). DOI 10.1007/978-3-642-20465-4_5
3. Aranha, D.F., Oliveira, L.B., López, J., Dahab, R.: Efficient implementation of elliptic curve cryptography in wireless sensors. Advances in Mathematics of Communications **4**(2), 169–187 (2010)
4. Arène, C., Lange, T., Naehrig, M., Ritzenthaler, C.: Faster computation of the Tate pairing. Journal of Number Theory **131**(5), 842–857 (2011). DOI 10.1016/j.jnt.2010.05.013
5. Barreto, P.S.L.M., Galbraith, S., Ó hÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography **42**(3), 239–271 (2007)
6. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Selected Areas in Cryptography, *Lecture Notes in Computer Science*, vol. 3897, pp. 319–331. Springer Berlin / Heidelberg (2006)
7. Bernstein, D.: A software implementation of NIST P-224. In: Presentation at the 5th Workshop on Elliptic Curve Cryptography (ECC 2001) (2001)
8. Bernstein, D., Lange, T.: Faster addition and doubling on elliptic curves. In: Advances in Cryptology — ASIACRYPT 2007, *Lecture Notes in Computer Science*, vol. 4833, pp. 29–50. Springer Berlin / Heidelberg (2008). DOI 10.1007/978-3-540-76900-2_3
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**(4), 297–319 (2004)
10. Certicom Research: SEC 2: Recommended elliptic curve domain parameters version 1.0 (2000). `http://www.secg.org/`
11. Comba, P.G.: Exponentiation cryptosystems on the IBM PC. IBM Systems Journal **29**(4), 526–538 (1990)
12. Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Transactions on Information Theory **30**(4), 587–594 (1984)
13. Dudacek, K., Vavricka, V.: Experimental evaluation of the MSP430 microcontroller power requirements. In: The International Conference on "Computer as a Tool" — EUROCON, 2007, pp. 400–404 (2007)
14. Dupont, R., Enge, A.: Provably secure non-interactive key distribution based on pairings. Discrete Applied Mathematics **154**(2), 270–276 (2006)
15. Fong, K., Hankerson, D., López, J., Menezes, A.: Field inversion and point halving revisited. IEEE Transactions on Computers **53**(8), 1047–1059 (2004)
16. Fuentes-Castañeda, L., Knapp, E., Rodríguez-Henríquez, F.: Faster hashing to $\mathbb{G}_2$. In: Selected Areas in Cryptography — SAC 2011 (2011)

17. Galindo, D., Roman, R., Lopez, J.: A killer application for pairings: Authenticated key establishment in underwater wireless sensor networks. In: Cryptology and Network Security, *Lecture Notes in Computer Science*, vol. 5339, pp. 120–132. Springer Berlin / Heidelberg (2008)
18. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Advances in Cryptology — CRYPTO 2001, *Lecture Notes in Computer Science*, vol. 2139, pp. 190–200. Springer Berlin / Heidelberg (2001)
19. Gouvêa, C.P.L., López, J.: Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller. In: Progress in Cryptology — INDOCRYPT 2009, *Lecture Notes in Computer Science*, vol. 5922, pp. 248–262. Springer Berlin / Heidelberg (2009)
20. Granger, R., Scott, M.: Faster squaring in the cyclotomic subgroup of sixth degree extensions. In: Public Key Cryptography — PKC 2010, *Lecture Notes in Computer Science*, vol. 6056, pp. 209–223. Springer Berlin / Heidelberg (2010)
21. Großschädl, J.: TinySA: A security architecture for wireless sensor networks. In: Proceedings of the 2006 ACM CoNEXT conference, p. 55. ACM New York (2006)
22. Guajardo, J., Blümel, R., Krieger, U., Paar, C.: Efficient implementation of elliptic curve cryptosystems on the TI MSP430x33x family of microcontrollers. In: Public Key Cryptography, *Lecture Notes in Computer Science*, vol. 1992, pp. 365–382. Springer Berlin / Heidelberg (2001)
23. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag New York (2004)
24. Karabina, K.: Squaring in cyclotomic subgroups. Cryptology ePrint Archive, Report 2010/542 (2010). `http://eprint.iacr.org/`
25. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics Doklady **7**, 595 (1963)
26. Knežević, M., Vercauteren, F., Verbauwhede, I.: Faster interleaved modular multiplication based on Barrett and Montgomery reduction methods. IEEE Transactions on Computers **59**(12), 1715–1721 (2010)
27. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography **28**, 119–134 (2003)
28. Lim, C.H., Lee, P.J.: More flexible exponentiation with precomputation. In: Advances in Cryptology — CRYPTO'94, *Lecture Notes in Computer Science*, vol. 839, pp. 95–107. Springer Berlin / Heidelberg (1994)
29. López, J., Dahab, R.: High-speed software multiplication in $\mathbb{F}_{2^m}$. In: Progress in Cryptology — INDOCRYPT 2000, *Lecture Notes in Computer Science*, vol. 1977, pp. 93–102. Springer Berlin / Heidelberg (2000)
30. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology **17**, 235–261 (2004)
31. Möller, B.: Algorithms for multi-exponentiation. In: Selected Areas in Cryptography, *Lecture Notes in Computer Science*, vol. 2259, pp. 165–180. Springer Berlin / Heidelberg (2001)
32. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation **44**(170), 519–521 (1985)
33. National Institute of Standards and Technology: Recommendation for key management (2007). `http://www.itl.nist.gov`
34. National Institute of Standards and Technology: FIPS 186-3: Digital signature standard (DSS) (2009). `http://www.itl.nist.gov`
35. Nogami, Y., Akane, M., Sakemi, Y., Kato, H., Morikawa, Y.: Integer variable χ-based Ate pairing. In: Pairing-Based Cryptography — Pairing 2008, *Lecture Notes in Computer Science*, vol. 5209, pp. 178–191. Springer Berlin / Heidelberg (2008)
36. Oliveira, L.B., Aranha, D.F., Gouvêa, C.P.L., Scott, M., Câmara, D.F., López, J., Dahab, R.: TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. Computer Communications **34**(3), 485–493 (2010)
37. Oliveira, L.B., Kansal, A., Gouvêa, C.P.L., Aranha, D.F., López, J., Priyantha, B., Goraczko, M., Zhao, F.: Secure-TWS: Authenticating node to multi-user communication in shared sensor networks. The Computer Journal (2011)
38. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan (2000)
39. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology **4**(3), 161–174 (1991)
40. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the final exponentiation for calculating pairings on ordinary elliptic curves. In: Pairing-Based Cryptography — Pairing 2009, *Lecture Notes in Computer Science*, vol. 5671. Springer Berlin / Heidelberg (2009)
41. Scott, M., Szczechowiak, P.: Optimizing multiprecision multiplication for public key cryptography. Cryptology ePrint Archive, Report 2007/299 (2007). `http://eprint.iacr.org/`
42. Solinas, J.A.: Efficient arithmetic on Koblitz curves. Designs, Codes and Cryptography **19**(2), 195–249 (2000)
43. Szczechowiak, P., Kargl, A., Scott, M., Collier, M.: On the application of pairing based cryptography to wireless sensor networks. In: Proceedings of the second ACM conference on Wireless network security, pp. 1–12. ACM New York (2009)
44. Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.: NanoECC: Testing the limits of elliptic curve cryptography in sensor networks. In: Wireless Sensor Networks, *Lecture Notes in Computer Science*, vol. 4913. Springer Berlin / Heidelberg (2008)
45. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (2010)
46. Weber, D., Denny, T.: The solution of McCurley's discrete log challenge. In: Advances in Cryptology — CRYPTO '98, *Lecture Notes in Computer Science*, vol. 1462, pp. 458–471. Springer Berlin / Heidelberg (1998). DOI 10.1007/BFb0055747
47. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Public Key Cryptography — PKC 2004, *Lecture Notes in Computer Science*, vol. 2947, pp. 277–290. Springer Berlin / Heidelberg (2004)