J Cryptol (2019) 32:867–894
https://doi.org/10.1007/s00145-018-9294-z

Koblitz Curves over Quadratic Fields

Thomaz Oliveira
Computer Science Department, CINVESTAV-IPN, Mexico City, Mexico
thomaz.figueiredo@gmail.com

Julio López
Institute of Computing, University of Campinas, Campinas, Brazil
jlopez@ic.unicamp.br

Daniel Cervantes-Vázquez · Francisco Rodríguez-Henríquez
Computer Science Department, CINVESTAV-IPN, Mexico City, Mexico
dcervantes@computacion.cs.cinvestav.mx
francisco@cs.cinvestav.mx

Communicated by François-Xavier Standaert.

Received 21 December 2016 / Revised 28 March 2018
Online publication 25 April 2018

Abstract. In this work, we retake an old idea that Koblitz presented in his landmark paper (Koblitz, in: Proceedings of CRYPTO 1991. LNCS, vol 576, Springer, Berlin, pp 279–287, 1991), where he suggested the possibility of defining anomalous elliptic curves over the base field $\mathbb{F}_4$. We present a careful implementation of the base and quadratic field arithmetic required for computing the scalar multiplication operation in such curves. We also introduce two ordinary Koblitz-like elliptic curves defined over $\mathbb{F}_4$ that are equipped with efficient endomorphisms. To the best of our knowledge, these endomorphisms have not been reported before. In order to achieve a fast reduction procedure, we adopted a redundant trinomial strategy that embeds elements of the field $\mathbb{F}_{4^m}$, with m a prime number, into a ring of higher order defined by an almost irreducible trinomial. We also suggest a number of techniques that allow us to take full advantage of the native vector instructions of high-end microprocessors. Our software library achieves the fastest timings reported for the computation of the timing-protected scalar multiplication on Koblitz curves, and competitive timings with respect to the speed records established recently in the computation of the scalar multiplication over binary and prime fields.

Keywords. Public-key cryptography, Elliptic curve cryptosystem, Implementation.

Thomaz Oliveira, Daniel Cervantes-Vázquez and Francisco Rodríguez-Henríquez would like to thank CONACyT (Project Number 180421) for their funding of this research. Julio López was supported in part by the Intel Labs University Research Office and by a research productivity scholarship from CNPq Brazil.

1. Introduction

Let $\mathbb{G}$ denote a cyclic group of order ℓ. Given an element $g \in \mathbb{G}$ of order $r|\ell$ and $h \in \langle g \rangle$, the Discrete Logarithm Problem (DLP) for $\mathbb{G}$ is the computational problem of finding an integer x such that $g^x = h$. The smallest integer x satisfying $g^x = h$ is called the discrete logarithm of h to the base g and is denoted by $\log_g h$. In this paper, we are mainly interested in the case where $\mathbb{G}$ is the additive group of points on an elliptic curve E defined over a binary field, and g is a point $P \in E(\mathbb{F}_{2^m})$.

In 1985, Koblitz [51] and Miller [62] independently showed that the group of points on an elliptic curve defined over a finite field could be used for designing a public key cryptosystem, having the DLP in that group as the underlying hard computational problem. This was the birth of Elliptic Curve Cryptography (ECC), which across the years has become one of the most intensively analyzed public key schemes in our discipline.[1]

In the abstract of his seminal paper, Koblitz [51] remarked that

> These elliptic curve cryptosystems may be more secure, because the analog of the discrete logarithm problem on elliptic curves is likely to be harder than the classical discrete logarithm problem, especially over $GF(2^n)$.

Indeed, it was soon realized that for sensible choices of the elliptic curve parameters, there did not appear to exist a subexponential-time algorithm that could solve the DLP over the group of points of an elliptic curve, or at least, one that was analogous to the index-calculus family of algorithms, which were proved to be highly successful when the DLP was defined in the multiplicative group of a finite field.

Furthermore, in the abstract of his paper [62], Miller commented about the disparity between the security offered by the original Diffie–Hellman protocol, whose security guarantees lie on the DLP defined over finite fields, as compared with its analogue over elliptic curves,

> As computational power grows, this disparity should get rapidly bigger.

As an interesting historical remark, let us recall that just 1 year before Koblitz and Miller presented their ECC proposal, Coppersmith had reported in [23][2] an index-calculus algorithm able to solve the DLP over characteristic two fields of the form $\mathbb{F}_q$ with $q = 2^m$, with a running time of $L_q(1/3, (32/9)^{\frac{1}{3}})$, where $L_q(\alpha, c)$, with parameters $0 < \alpha < 1$ and $c > 0$, denotes the expression, $\exp\left((c + o(1))(\log q)^{\alpha}(\log\log q)^{1-\alpha}\right)$. Almost 30 years after, in February 2013, Joux [47] presented a new DLP algorithm with a running time of $L_q(1/4, c)$ (for some undetermined c), which was rapidly followed by several other developments that culminated with the discovery of algorithms that asymptotically enjoy a quasi-polynomial time complexity [6,38]. Hence, a bit less than three decades of cryptanalysis rendered the DLP in binary fields completely useless for constructive cryptographic purposes.

In stark contrast with its binary field instantiation, and after all these years of far and wide analysis, the DLP over elliptic curves defined in the binary field $\mathbb{F}_{q=2^m}$, still stands as a formidable computational task. As of today, the best known general-purpose

[1] See [54] for a historical recount of the first three decades of elliptic curve cryptography.

[2] Building on the work by Blake et al. [16].

algorithm to solve it is still the Pollard rho algorithm [30]. In the last decade, however, inspired by the new lines of research presented by Semaev [79], several researchers [29, 31,35,49,80] (see also [30] for a comprehensive survey) have attempted to attack the DLP of all binary elliptic curves using summation-polynomial methods.[3] However, the current status of these attempts is crucially based on ill-understood Gröbner basis assumptions, which in some cases have led to contradictory behaviors [42], even for tiny experiments [30].[4]

On the other side, it is now standard knowledge that the Pollard rho algorithm is able to solve the DLP over generic curves with an exponential computational complexity of $(1 + o(1))O(\frac{\sqrt{\pi \cdot q}}{2})$. Moreover, it is always possible to apply the *negation map*, which yields an extra $\sqrt{2} - o(1) < 1.5$ improvement to the above estimate [10,90]. Further, in the case of Koblitz curves, which are the main subject of this paper, one can apply the Frobenius endomorphism to speed up the Pollard rho algorithm by an extra $\sqrt{m}$ factor [34,91]. Notice that in practice this $\sqrt{m}$ factor implies a relatively modest computational saving of no more than 4–5 bits. Taking into account the above, several international bodies have suggested that the size of the binary field where cryptographic elliptic curves are defined should be designed by "adding about 10 bits in the binary field case" [27].[5]

1.1. *Known Attacks Against Certain Classes of Binary Curves*

At the time that ECC was proposed, it appeared clear that binary supersingular curves were the most efficient curves in practice [52,61]. However, when Menezes, Okamoto, and Vanstone published their famous MOV attack [58,59], it was realized that the difficulty of computing discrete logarithms in $E(\mathbb{F}_{2^m})$ is actually comparable to the security provided by the DLP in the multiplicative group of the field $\mathbb{F}_{2^{2m}}$. Because of this, it was recommended not to use any kind of supersingular curves for cryptographic applications.

Surprisingly, this situation changed around the year 2000, when several papers proposing the usage of pairing-based protocols were published [19,45,46,76]. All of a sudden, and after having been banned for almost a decade, it was again a good idea to implement cryptographic applications using binary (and also ternary) supersingular curves. As a consequence, several works reported efficient implementations using those curves both in hardware and in software platforms [4,13–15]. Nevertheless, disaster struck again about one decade later, when Joux [47] discovered his aforementioned DLP algorithm for binary field multiplicative groups, with a running time of $L_q(1/4, c)$, for $q = 2^m$, $c > 0$. This marked the end of the usage of binary supersingular curves in cryptography.

Given a target ordinary binary curve defined over the field $\mathbb{F}_{2^m}$, the Gaudry–Hess–Smart (GHS) attack [33,36,41,60] exploits the idea of finding an algebraic curve C of a relatively small genus g such that the Jacobian of C contains the target elliptic

[3] Sometimes also called Semaev's polynomials.

[4] It is interesting to note that Semaev's original work in [79] described an attack that in principle can be applied not only to binary but to all elliptic curves [75].

[5] Nevertheless, the influential French Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) considers in [2] that in terms of their security, "Les courbes elliptiques définies sur $GF(p)$ ne sont pas différenciées de celles définies sur $GF(2^n)$" (elliptic curves defined over $GF(p)$ should not be differentiated from those defined over $GF(2^n)$).

curve group. In this case, the original elliptic curve discrete logarithm problem can be transferred into the Jacobian of C defined over $\mathbb{F}_{2^l}$, with $l|m$. The hope is that if the genus of C is not too large, the DLP could be easier to solve in that Jacobian, due to the availability of an index-calculus strategy for that group [28].

In general, however, the GHS strategy is difficult to implement due to the large genus that a suitable curve C usually ends up having.[6] In fact, in [36,60], it was proved that the GHS attack fails (i.e., the Pollard rho attack is more effective), for all binary elliptic curves defined over $\mathbb{F}_{2^m}$, where $m \in [160, 600]$ is prime. Furthermore, in [57], it was proved that the GHS attack fails for most of the composite extensions in the range $m \in [160, 600]$. To our knowledge, the largest instance where the GHS has been proved effective is for the DLP computation over $E(\mathbb{F}_{2^{5 \cdot 31}})$. In [87], it was estimated that when using the Enge–Gaudry algorithm [28], the cost of such computation is of around 1736 core days.

1.2. *Performance Advantage of Ordinary Binary Elliptic Curves*

The computation of the scalar multiplication operation on binary elliptic curves can be performed significantly faster than prime field curves for both software and hardware platforms.

In software, the libraries reported in [3,70,71,84] rank among the fastest Diffie–Hellman software benchmarked in the eBACS site [11] in various platforms. In particular, the software library announced in [70] holds the current speed record for constant-time variable-base-point Diffie–Hellman software at the 128 bit security level.

In hardware, due to the carry-less arithmetic, the computation of the scalar multiplication operation on binary elliptic curves implies a substantial hardware circuitry saving when compared to the full adders and integer multipliers required for prime field elliptic curves. Consequently, ECC accelerators using binary curves tend to be more compact and faster than their prime field curve counterparts (see [5, Table IV] for a comparison of recent designs). It should be remarked that for the Internet of Things and several other applications, hardware performance is much more important than software performance.

1.3. *Choosing Side Channel-Resistant Binary Elliptic Curves*

Since the publication of Kocher's paper on differential power analysis [55], the cryptographic community has been increasingly concerned about the importance of producing side channel-resistant cryptographic software and hardware. In a post-Snowden world, this fear has just exacerbated. In the case of ECC, one first line of defense is the sound selection of elliptic curves that show solid cryptographic properties against several known side-channel attacks, which apply both in hardware and in software platforms.

In [12] (see also [20]), the authors present several criteria that a so-called *safe curve* should exhibit. Among others, the following properties are listed: rigidity, safety against transfers, Montgomery-ladder-friendliness, twist security, etc. The reader is referred to [12] for a definition of each one of the desirable security properties that a safe curve should enjoy. Unfortunately, the authors in [12,20] explicitly exclude binary elliptic curves from their analysis, without elaborating in the technical arguments to avoid them.

[6]Another problem may occur if the genus of C is too small. For example, the Jacobian of a curve C in $\mathbb{F}_2$ would be too small to give any useful information about the DLP over $E(\mathbb{F}_{2^m})$ [57].

However, and as it will be discussed in the remaining of this paper, Koblitz curves (see their definition in the next subsection) meet most of the requirements specified for safe curves in [12]. Particularly, Koblitz curves arguably stand among the purest mathematical elliptic curve problems, which makes them as *fully rigid* as it gets. Similarly, these curves support the most efficient Montgomery ladder formulas that we know of (consisting of only five field multiplications per bit, which have the extra bonus of being amenable for parallelization). Moreover, Koblitz curves are suitable for right-to-left Montgomery ladders as discussed in [69]. This feature is especially valuable for the vast majority of protocols (such as the Diffie–Hellman protocol, digital signatures, key generation, etc.), which usually require the computation of one or more fixed-point scalar multiplications. Similarly, Koblitz curves enjoy *twist security* and they are *transfer safe*. On the other hand, they do not meet the *completeness* criterion as defined in [12].

1.4. *Koblitz Curves*

Anomalous binary curves, generally referred to as Koblitz curves, are binary elliptic curves satisfying the Weierstrass equation, $E_a : y^2+xy = x^3+ax^2+1$, with $a \in \{0, 1\}$. Since their introduction in 1991 by Koblitz [53], these curves have been extensively studied for their additional structure that allows, in principle, a performance speedup in the computation of the elliptic curve point multiplication operation. Also, Koblitz curves were historically the first family of ordinary elliptic curves proposed for cryptographic usage [52].

Koblitz curves defined over $\mathbb{F}_4$ were also proposed in [53]. Nevertheless, until now the research works dealing with standardized Koblitz curves in commercial use, such as the binary curves standardized by NIST [65–67] or the suite of elliptic curves supported by the TLS protocol [17,25], have exclusively analyzed the security and performance of curves defined over binary extension fields $\mathbb{F}_{2^m}$, with m a prime number (for recent examples, see [3,18,84,89]).

We found interesting to explore the cryptographic usage of Koblitz curves defined over $\mathbb{F}_4$ due to their inherent usage of quadratic field arithmetic. Indeed, it has been recently shown [56,69,71] that quadratic field arithmetic is extraordinarily efficient when implemented in software. This is because one can take full advantage of the Single Instruction Multiple Data (SIMD) paradigm, where a vector instruction performs simultaneously the same operation on a set of input data items.

Quadratic extensions of a binary finite field $\mathbb{F}_{q^2}$ can be defined by means of a monic polynomial of degree two $h(u) \in \mathbb{F}_2[u]$ irreducible over $\mathbb{F}_q$. The field $\mathbb{F}_{q^2}$ is isomorphic to $\mathbb{F}_q[u]/(h(u))$, and its elements can be represented as $a_0 + a_1u$, with $a_0, a_1 \in \mathbb{F}_q$. The addition of two elements $a, b \in \mathbb{F}_{q^2}$, can be performed as $c = (a_0 + b_0) + (a_1 + b_1)u$. Using $h(u) = u^2 + u + 1$, the multiplication of a, b can be computed as $d = a_0b_0 + a_1b_1 + ((a_0 + a_1) \cdot (b_0 + b_1) + a_0b_0)u$. By carefully organizing the code associated with these arithmetic operations, one can greatly exploit the instruction-level parallelism of the pipelines that are available in contemporary high-end processors.

1.4.1. *Our Contributions*

In this work, we designed for the first time a 128-bit secure and timing attack-resistant scalar multiplication on a Koblitz curve defined over $\mathbb{F}_4$, as it was proposed in [53].

Furthermore, we present a taxonomy of Koblitz-like elliptic curves. Some of these curves are equipped with more efficient endomorphisms, which to the best of our knowledge have not been discussed before.

We developed all the required algorithms for performing the scalar multiplication at the 128-bit security level for standard Koblitz curves and also for one of the Koblitz-like elliptic curves introduced in this paper for the first time. This took us to reconsider the strategy of using redundant trinomials (also known as almost irreducible trinomials), which were proposed more than 10 years ago in [21,26]. We also report what is perhaps the most comprehensive analysis yet reported on how to efficiently implement arithmetic operations in binary finite fields and their quadratic extensions using the vectorized instructions available in high-end microprocessors.

The remaining of this paper is organized as follows. In Sect. 2, we formally introduce the family of Koblitz elliptic curves defined over $\mathbb{F}_4$. In Sect. 3, we analyze all the 12 ordinary elliptic curves that can be defined over $\mathbb{F}_4$ and their classification in isogeny classes. For one isogeny class, curves equipped with efficient endomorphisms are introduced. In Sects. 4 and 5, a detailed description of the efficient implementation of the base and quadratic field arithmetic using vectorized instructions is given. We introduce in Sect. 6 the scalar multiplication algorithms used in this work, and we present in Sect. 7 the analysis and discussion of the results obtained by our software library. Finally, we draw our concluding remarks in Sect. 8.

2. Koblitz Curves over $\mathbb{F}_4$

Koblitz curves over $\mathbb{F}_4$ are defined by the following equation

$$E_a : y^2 + xy = x^3 + a\gamma x^2 + \gamma, \tag{1}$$

where $\gamma \in \mathbb{F}_4$ satisfies $\gamma^2 = \gamma + 1$ and $a \in \{0, 1\}$. The number of points in the curves $E_0/\mathbb{F}_4$ and $E_1/\mathbb{F}_4$ is 4 and 6, respectively. For cryptographic purposes, one uses (1) operating over binary extension fields of the form $\mathbb{F}_{4^m}$, and m is a prime number. The set of affine points $P = (x, y) \in \mathbb{F}_{4^m} \times \mathbb{F}_{4^m}$ that satisfy (1) together with a point at infinity represented as $\mathcal{O}$ forms an abelian group denoted by $E_a(\mathbb{F}_{4^m})$, where its group law is defined by the point addition operation.

For each proper divisor r of s, $E(\mathbb{F}_{4^r})$ is a subgroup of $E(\mathbb{F}_{4^s})$, then $\#E(\mathbb{F}_{4^r})$ divides $\#E(\mathbb{F}_{4^s})$ and consequently, the order of every group of points on a quadratic Koblitz curve is divisible by 4 or 6 (depending on the curve a-parameter). On the other hand, by choosing specific prime extensions m, it is possible to find groups $E_a(\mathbb{F}_{4^m})$ with almost-prime order, for instance, $E_0(\mathbb{F}_{4^{163}})$ and $E_1(\mathbb{F}_{4^{167}})$. In Table 1, we present the almost-prime group orders $\#E_a(\mathbb{F}_{4^m})$ for prime degrees $m \in \{127, \ldots, 191\}$.

The Frobenius map $\tau : E_a(\mathbb{F}_{4^m}) \rightarrow E_a(\mathbb{F}_{4^m})$ defined by $\tau(\mathcal{O}) = \mathcal{O}$, $\tau(x, y) = (x^4, y^4)$, is a curve automorphism satisfying $(\tau^2 + 4)P = \mu\tau(P)$ for $\mu = (-1)^a$ and all $P \in E_a(\mathbb{F}_{4^m})$. By solving the equation $\tau^2 + 4 = \mu\tau$, the Frobenius map can be seen as the complex number $\tau = (\mu \pm \sqrt{-15})/2$.

Table 1. Almost-prime group orders $\#E_a(\mathbb{F}_{4^m})$ with prime $m \in \{127, \ldots, 191\}$.

m	a	Factorization of $\#E_a(\mathbb{F}_{4^m})$
127	0	$0x4 \cdot \underline{0x1268F1298760419} \cdot \underline{0xDE7D169BED4130151CD618CF571307} \underline{7271FF51A4B1CFB75BF}$ (196)
127	1	$0x6 \cdot 0x41603EAF071 \cdot \underline{0x29C4C778B6D2CD0FA36B3CA951A32} \underline{DAC100C9C63576EEF7BF1F21}$ (209)
131	1	$0x6 \cdot 0x4267F1026F4F \cdot \underline{0x2806BB97FB5F7C2F9E1EDE20BF59AC39} \underline{0DABBA7621D9A0F26AA1}$ (205)
137	0	$0x4 \cdot 0x763DB379950B73D200B971F1D \cdot \underline{0x22A41FB03F2428B44188} \underline{DD9FFEA796DC6D197A91BA21}$ (173)
137	1	$0x6 \cdot 0x4337925B3141B99447C1273 \cdot \underline{0x289FE5979AC03A2E5} \underline{CFCE8E6024FEF0863C633AE96A0DF}$ (182)
149	0	$0x4 \cdot 0x29B66B578C9FAEB \cdot \underline{0x62322066993B57A8857E552587C80A5} \underline{67018483F2E493DBB7750AB7DB623}$ (239)
149	1	$0x6 \cdot \underline{0x1B73C442E8D} \cdot \underline{0x637845F7F8BFAB325B85412FB} \underline{54061F148B7F6E79AE11CC843ADE1470F7E4E29}$ (255)
157	0	$0x4 \cdot 0x499D09449B55C7D71FC18A2B0265785F \cdot \underline{0x37A45BD5E114A84FCB8} \underline{900BAEA9E731E0C4B3EDEC15F327}$ (186)
157	1	$0x6 \cdot 0xEECA8C4698A0916800B4E7 \cdot \underline{0xB6F74A858FF10701D113} \underline{E39259417F04CF038B297F3C6573F6E14F33}$ (224)
163	0	$0x4 \cdot \underline{0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF}\backslash \underline{EA48D724AAB2045E5CFE286F8372017024DFF7BB3}$(324)
167	1	$0x6 \cdot \underline{0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA}\backslash \underline{D45C6A4A8565763007E9FEFA42E0EA9B9E8B7F3541}$ (331)
173	1	$0x6 \cdot 0xBA3DEF139 \cdot \underline{0xEA9746EEF14E1638A503F} \underline{A6FB739A623894A590811B6939A30D7A016E8A77815}\backslash\ \underline{0084D9C4D6E0D}$ (308)
179	0	$0x4 \cdot 0x10C01861F3F8F0AC2767CD \cdot \underline{0xF4882969C296A9493FEAA3C9F58} \underline{DA166B76D3236BF15C2F10E2B0421F3F7E50DCC6F}$ (272)
181	0	$0x4 \cdot 0xCBB \cdot \underline{0x141BF6E35420FDE10CF60620853943A20D5} \underline{A91F2F5DDE75B04126F3100B191AF1}\backslash\ \underline{E338F81FB8ED77C1C57BEF3}$ (348)
191	1	$0x6 \cdot 0x23D01 \cdot \underline{0x4C3F9B376D369D04F034} \underline{99007A43FE6460A012C86B2C575}\ \underline{858EE9FC7F67A566813}\backslash \underline{B39DA28DC9D58285BC07F8811}$ (362)

Prime factors are underlined. The size (in bits) of the largest prime factor is presented in parenthesis

2.1. *The τ-Adic Representation*

Given a Koblitz curve $E_a/\mathbb{F}_{4^m}$ with group order $\#E_a(\mathbb{F}_{4^m}) = h \cdot p \cdot r$, where h is the order of $E_a(\mathbb{F}_4)$, r is the order of our subgroup of interest, and p is the order of a group of no cryptographic interest,[7] we can express a scalar $k \in \mathbb{Z}/r\mathbb{Z}$ as an element in $\mathbb{Z}[\tau]$ using the classical partial reduction introduced by Solinas [82], with a few modifications. The modified version is based on the fact that $\tau^2 = \mu\tau - 4$.

Given that the norm of τ is $N(\tau) = 4$, $N(\tau - 1) = h$, $N(\tau^m - 1) = h \cdot p \cdot r$, and $N((\tau^m - 1)/(\tau - 1)) = p \cdot r$, the subscalars k_0 and k_1 resulted from the partial modulo function will be both of size approximately $\sqrt{p \cdot r}$. As a consequence, the corresponding scalar multiplication will need more iterations than expected, since the

[7] Usually the order p is composite. Also, every prime factor of p is smaller than r (see Table 1).

order p of a subgroup which is not of cryptographic interest will also be taken into account in the computation. For that reason, we took the design decision of considering that the input scalar of our point multiplication algorithm is already given in the $\mathbb{Z}[\tau]$ domain.

As a result, a partial reduction of the scalar k is no longer required, and the number of iterations in the point multiplication will be consistent with the scalar k size. If one needs to retrieve the equivalent value of the scalar k in the ring $\mathbb{Z}/r\mathbb{Z}$, this can be easily computed with one multiplication and one addition in $\mathbb{Z}/r\mathbb{Z}$. This strategy is in line with the degree-2 scalar decomposition method within the GLS curves context as suggested in [32].

2.2. *The Width-w τNAF Form*

Assuming that the scalar k is specified in the $\mathbb{Z}[\tau]$ domain, one can represent the scalar in the regular width-w τNAF form [69] by slightly adopting the method for the $\mathbb{F}_4$ case. The length of the representation width-w τNAF of an element $k \in \mathbb{Z}[\tau]$ is discussed in [81]. Given a width w, after running the regular τNAF algorithm, we have $2^{2(w-1)-1}$ different digits.[8]

Therefore, it is necessary to be more conservative when choosing the width w, when compared to the Koblitz curves defined over $\mathbb{F}_2$. For widths $w = 2, 3, 4, 5$, we have to pre- or post-compute 2, 8, 32, and 128 points, respectively. In order to construct an efficient 128-bit point multiplication, we estimated that the value of the width w must be at most four. Otherwise, the costs of the point pre- and post-processing will be greater than the additional savings obtained in the main iteration.

In addition, we must find efficient ways for computing the values $\alpha_v = v \bmod \tau^w$. The method for searching the best expressions in Koblitz curves over $\mathbb{F}_2$ [85] cannot be directly applied in the $\mathbb{F}_4$ case. As a result, we manually provided α_v representations for $w \in \{2, 3, 4\}$ and $a = 1$, which are our implementation parameters.[9] The rationale for our chosen representations was to minimize the number of field arithmetic operations. In practice, we must reduce the number of full point additions on behalf of point doublings and mixed additions.[10] In Table 2, we present the α_v representatives along with the operations required to generate the multiples of the base point.[11]

As a result, one point doubling and full addition are required to generate the points $\alpha_v P$ for $w = 2$, one point doubling, four full additions, three mixed additions, and four applications of the Frobenius map for the $w = 3$ case and one point doubling, twenty full additions, eleven mixed additions, and five applications of the Frobenius map for the $w = 4$ case.

[8] We are considering only positive digits, since the cost of computing the negative points in binary elliptic curves is negligible.

[9] While this is undoubtedly not the optimal approach for computing these values, the point pre-computation represents only a tiny fraction of the whole scalar multiplication performance cost.

[10] Full addition is the operation of adding two points both represented in projective coordinates, whereas a mixed addition is the operation of adding one point represented in projective coordinates with another represented in affine coordinates. Note that a full addition is always more costly than a mixed addition.

[11] Notice that the multiples $\alpha_v P$ as shown in Table 2 must be computed out of order. The order for computing the multiples is shown in roman numbers.

Table 2. Representations of $\alpha_v = v \bmod \tau^w$, for $w \in \{2, 3, 4\}$, $a = 1$ and the required operations for computing α_v.

w	v	$v \bmod \tau^w$	α_v	Operations	Order
2	1	1	1	n/a	I
	3	3	3	$t_0 \leftarrow 2\alpha_1, \alpha_3 \leftarrow t_0 + \alpha_1$ $(D + FA)$	II
3	1	1	1	n/a	I
	3	3	3	$t_0 \leftarrow 2\alpha_1, \alpha_3 \leftarrow t_0 + \alpha_1$ $(D + FA)$	II
	5	5	$-\tau - \alpha_{15}$	$\alpha_5 \leftarrow -t_1 - \alpha_{15}$ (MA)	VIII
	7	$3\tau + 3$	$\tau^2\alpha_3 + \alpha_3$	$\alpha_7 \leftarrow \tau^2\alpha_3 + \alpha_3$ $(FA + 2T)$	III
	9	$3\tau + 5$	$\alpha_7 + 2$	$\alpha_9 \leftarrow \alpha_7 + t_0$ (FA)	IV
	11	$3\tau + 7$	$\alpha_9 + 2$	$\alpha_{11} \leftarrow \alpha_9 + t_0$ (FA)	V
	13	$-\tau - 7$	$\tau^2 - \alpha_3$	$\alpha_{13} \leftarrow t_2 - \alpha_3$ (MA)	VII
	15	$-\tau - 5$	$\tau^2 - 1$	$t_1 \leftarrow \tau\alpha_1, t_2 \leftarrow \tau t_1, \alpha_{15} \leftarrow t_2 - \alpha_1$ $(MA + 2T)$	VI
4	1	1	1	n/a	I
	3	3	$-\tau^3 - \alpha_{61}$	$\alpha_3 \leftarrow -t_4 - \alpha_{61}$ (MA)	XXVI
	5	5	$-\tau^3 - \alpha_{59}$	$\alpha_5 \leftarrow -t_4 - \alpha_{59}$ (MA)	XXVII
	7	7	$-\tau^3 - \alpha_{57}$	$\alpha_7 \leftarrow -t_4 - \alpha_{57}$ (MA)	XXVIII
	9	9	$-\tau^3 - \alpha_{55}$	$\alpha_9 \leftarrow -t_4 - \alpha_{55}$ (MA)	XXIX
	11	11	$-2\tau^2 + \alpha_{43}$	$\alpha_{11} \leftarrow -t_2 + \alpha_{43}$ (FA)	XXX
	13	13	$-2\tau^2 + \alpha_{45}$	$\alpha_{13} \leftarrow -t_2 + \alpha_{45}$ (FA)	XXXI
	15	15	$-2\tau^2 + \alpha_{47}$	$\alpha_{15} \leftarrow -t_2 + \alpha_{47}$ (FA)	XXXII
	17	$5\tau - 11$	$-\tau^3 - \alpha_{47}$	$t_4 \leftarrow \tau^2 t_3, \alpha_{17} \leftarrow -t_4 - \alpha_{47}$ $(MA + 2T)$	XIX
	19	$5\tau - 9$	$-\tau^3 - \alpha_{45}$	$\alpha_{17} \leftarrow -t_4 - \alpha_{47}$ (MA)	XX
	21	$5\tau - 7$	$-\tau^3 - \alpha_{43}$	$\alpha_{17} \leftarrow -t_4 - \alpha_{45}$ (MA)	XXI
	23	$5\tau - 5$	$-\tau^3 - \alpha_{41}$	$\alpha_{17} \leftarrow -t_4 - \alpha_{43}$ (MA)	XXII
	25	$5\tau - 3$	$-\tau^3 - \alpha_{39}$	$\alpha_{17} \leftarrow -t_4 - \alpha_{41}$ (MA)	XXIII
	27	$5\tau - 1$	$-\tau^3 - \alpha_{37}$	$\alpha_{17} \leftarrow -t_4 - \alpha_{39}$ (MA)	XXIV
	29	$5\tau + 1$	$-\tau^3 - \alpha_{35}$	$\alpha_{17} \leftarrow -t_4 - \alpha_{37}$ (MA)	XXV
	31	$-2\tau - 9$	$2\tau^2 - 1$	$t_2 \leftarrow \tau t_1, \alpha_{31} \leftarrow t_2 - \alpha_1$ $(MA + T)$	XII
	33	$-2\tau - 7$	$2\tau^2 + 1$	$\alpha_{33} \leftarrow t_2 + \alpha_1$ (MA)	XIII
	35	$-2\tau - 5$	$-2\tau - 5$	$\alpha_{35} \leftarrow \alpha_{37} - t_0$ (FA)	VI
	37	$-2\tau - 3$	$-2\tau - 3$	$\alpha_{37} \leftarrow \alpha_{39} - t_0$ (FA)	IV
	39	$-2\tau - 1$	$-2\tau - 1$	$t_0 \leftarrow 2\alpha_1, t_1 \leftarrow \tau t_0, \alpha_{39} \leftarrow -t_1 - \alpha_1$ $(D + MA + T)$	II
	41	$-2\tau + 1$	$-2\tau + 1$	$\alpha_{41} \leftarrow -t_1 + \alpha_1$ (MA)	III
	43	$-2\tau + 3$	$-2\tau + 3$	$\alpha_{43} \leftarrow \alpha_{41} + t_0$ (FA)	V
	45	$-2\tau + 5$	$-2\tau + 5$	$\alpha_{45} \leftarrow \alpha_{43} + t_0$ (FA)	VII
	47	$-2\tau + 7$	$-2\tau + 7$	$\alpha_{47} \leftarrow \alpha_{45} + t_0$ (FA)	VIII
	49	$-2\tau + 9$	$-2\tau + 9$	$\alpha_{49} \leftarrow \alpha_{47} + t_0$ (FA)	IX
	51	$-2\tau + 11$	$-2\tau + 11$	$\alpha_{51} \leftarrow \alpha_{49} + t_0$ (FA)	X
	53	$-2\tau + 13$	$-2\tau + 13$	$\alpha_{53} \leftarrow \alpha_{51} + t_0$ (FA)	XI
	55	$3\tau - 13$	$3\tau - 13$	$t_3 \leftarrow \tau\alpha_1, \alpha_{55} \leftarrow t_3 - \alpha_{53}$ $(MA + T)$	XIV
	57	$3\tau - 11$	$3\tau - 11$	$\alpha_{57} \leftarrow t_3 - \alpha_{51}$ (MA)	XV
	59	$3\tau - 9$	$3\tau - 9$	$\alpha_{59} \leftarrow t_3 - \alpha_{49}$ (MA)	XVI
	61	$3\tau - 7$	$3\tau - 7$	$\alpha_{61} \leftarrow t_3 - \alpha_{47}$ (MA)	XVII
	63	$3\tau - 5$	$3\tau - 5$	$\alpha_{63} \leftarrow t_3 - \alpha_{45}$ (MA)	XVIII

Here, we denote by D, FA, MA, T the point doubling, full addition, mixed addition, and the Frobenius map, respectively. Moreover, we consider that the point $\alpha_1 P$ is represented in affine coordinates. The order for computing the points is given in roman numbers

2.3. *Security Analysis Against the GHS Attack*

Given that the Koblitz curves defined over $E_a(\mathbb{F}_{4^m})$ operate over quadratic extensions fields, it is conceivable that Weil descent attacks [36,41] could possibly be efficiently applied on these curves. However, Menezes and Qu showed in [60] that the GHS attack cannot be implemented efficiently for elliptic curves defined over binary extension fields $\mathbb{F}_q$, with $q = 2^m$, and m a prime number in [160, 600]. Further, a specialized analysis for binary curves defined over fields of the form $\mathbb{F}_{4^m}$ reported in [39] proved that the only vulnerable prime extension in the range [80, 256] is $m = 127$.

3. Extended Koblitz Curves over $\mathbb{F}_4$

There exist several ordinary elliptic curves over $\mathbb{F}_4$ that strictly speaking cannot be considered Koblitz curves in the way that they were defined in Sect. 2. Since some of these additional curves come out equipped with additional endomorphisms, they are also of cryptographic interest. This *extended* set of Koblitz curves can be better described using isogeny classes as discussed in the following subsection.

3.1. *Isogenies*

Let E_1 and E_2 be two elliptic curves defined over a field $\mathbb{K}$. An isogeny map is a non-constant homomorphism $\phi : E_0(\bar{\mathbb{K}}) \to E_1(\bar{\mathbb{K}})$ such that $\phi(\mathcal{O}) = \mathcal{O}$, which can be described by means of rational functions. Moreover, $\phi(P + Q) = \phi(P) + \phi(Q)$ and $\phi(x_0, y_0) = (x_1, y_1)$, where $x_1 = R(x_0)$ and $y_1 = y_0 R(x_0)$ are rational functions. If $R(x) = \frac{p(x)}{q(x)}$ and $\gcd(p(x), q(x)) = 1$, then the degree of ϕ is $\max\{\deg(p(x)), \deg(q(x))\}$. If ϕ has the same domain and co-domain, then ϕ is an endomorphism. If ϕ has degree one, then ϕ is an isomorphism. Two elliptic curves are isomorphic if an isomorphism between them exists.

It is known that two curves E_0 and E_1 are isogenous over $\mathbb{K}$ if and only if they have the same number of points [83, Theorem 1]. This fact helps the elliptic curves classification by isogeny classes, that is, by their point cardinality.

3.2. *The Twelve Ordinary Elliptic Curves over* $\mathbb{F}_4$

The description of Koblitz curves given in Sect. 2 defines four ordinary elliptic curves. Nevertheless, there is a total of 12 ordinary elliptic curves over the field $\mathbb{F}_4$. As shown in Table 3, these 12 ordinary elliptic curves define four different isogeny classes.

Notice that the Koblitz curves described in Sect. 2 correspond to isogeny classes 1 and 2 of Table 3, with curve parameters (a, b) given as (γ, γ), (γ^2, γ^2), $(0, \gamma)$, and $(0, \gamma^2)$. Since these two classes were already studied, in the following, we will focus our attention on isogeny classes 0 and 3.

The curves $E_{(1,1)}$ and $E_{(0,1)}$ are the only two members of isogeny class 0. Notice that the curve parameters a, b of these curves lie in $\mathbb{F}_2$. Furthermore, it can be shown that $E_{(1,1)}$ and $E_{(0,1)}$ become isomorphic over $\mathbb{F}_4$ and that $(\#E_{(0,1)}(\mathbb{F}_2) \cdot \#E_{(1,1)}(\mathbb{F}_2)) \mid \#E_0(\mathbb{F}_4)$, where E_0 is the Koblitz curve defined in (1) with $a = 0$. This observation

Table 3. Twelve ordinary elliptic curves $E_{(a,b)}/\mathbb{F}_4 : y^2 + xy = x^3 + ax^2 + b$ define four isogeny classes. The curve parameters $a, b \in \mathbb{F}_4$ can take the values $[0, 1, \gamma, \gamma^2]$, with $\gamma \in \mathbb{F}_4 \backslash \mathbb{F}_2$.

	$E_{(a,b)}/\mathbb{F}_4 : y^2 + xy = x^3 + ax^2 + b$			
Isogeny class	0	1	2	3
$\#E_{(a,b)}$	8	6	4	2
Parameters (a, b)	$(1, 1)$, $(0, 1)$.	$(1, \gamma)$, $(1, \gamma^2)$, $(0, \gamma)$, $(0, \gamma^2)$.	(γ, γ), (γ^2, γ^2), (γ, γ^2), (γ^2, γ).	$(\gamma, 1)$, $(\gamma^2, 1)$.

can be generalized to prove that $(\#E_{(0,1)}(\mathbb{F}_{2^m}) \cdot \#E_{(1,1)}(\mathbb{F}_{2^m})) \mid \#E_0(\mathbb{F}_{4^m})$. A direct consequence of this relation is that the largest prime factor of $\#E_0(\mathbb{F}_{4^m})$ must be smaller than the order $\#E_{(0,1)}(\mathbb{F}_{2^m}) \approx \#E_{(1,1)}(\mathbb{F}_{2^m}) \approx 2^m$. Thus, when the two curves in the isogeny class 0 are defined over the field $\mathbb{F}_{4^m}$, one can only hope to achieve at most an $\frac{m}{2}$-bit security level. Hence, we conclude that isogeny class 0 is of little or no cryptographic value.

3.3. *A Novel Endomorphism for Isogeny Class 3*

The curves $E_{(\gamma,1)}$ and $E_{(\gamma^2,1)}$ are the only two members of isogeny class 3 shown in Table 3. Both of these two curves are equipped with an efficient endomorphism as discussed next.

It can be seen that the mapping $\tau_2 : E_{(a,b)}(\mathbb{F}_q) \rightarrow E_{(a',b')}(\mathbb{F}_q)$ defined by $\tau(\mathcal{O}) = \mathcal{O}$, $\tau(x, y) = (x^2, y^2)$ is a two-degree isogeny such that $\tau_2(E_{(\gamma,1)}(\mathbb{F}_q)) = E_{(\gamma^2,1)}(\mathbb{F}_q)$, and $\tau_2(E_{(\gamma^2,1)}(\mathbb{F}_q)) = E_{(\gamma,1)}(\mathbb{F}_q)$. Moreover, the curves $E_{(\gamma,1)}$ and $E_{(\gamma^2,1)}$ are also isomorphic, since one can define the isogenies, $\phi_0 : E_{(\gamma,1)} \rightarrow E_{(\gamma^2,1)}$ and $\phi_1 : E_{(\gamma^2,1)} \rightarrow E_{(\gamma,1)}$ such that $\phi_0(x, y) = (x, y+\gamma \cdot x)$ and $\phi_1(x, y) = (x, y+\gamma^2 \cdot x)$. As illustrated in Fig. 1, for each one of the curves in class 3, one can therefore build two novel endomorphisms $\bar{\tau}_0, \bar{\tau}_1$ as follows:

$$\bar{\tau}_0(x, y) = (\phi_0 \circ \tau_2)(x, y) = (x^2, y^2 + \gamma \cdot x^2)$$
$$\bar{\tau}_1(x, y) = (\phi_1 \circ \tau_2)(x, y) = (x^2, y^2 + \gamma^2 \cdot x^2).$$

Using affine λ-coordinates as defined in [71], the endomorphisms $\bar{\tau}_0$ and $\bar{\tau}_1$ can be written as $\bar{\tau}_0(x, \lambda) = (x^2, \lambda^2 + \gamma)$ and $\bar{\tau}_1(x, \lambda) = (x^2, \lambda^2 + \gamma^2)$, respectively. Using projective λ-coordinates, they become $\bar{\tau}_0(X, L, Z) = (X^2, L^2 + \gamma \cdot Z^2, Z^2)$ and $\bar{\tau}_1(X, L, Z) = (X^2, L^2 + \gamma^2 \cdot Z^2, Z^2)$.

Since $\bar{\tau}_0$ and $\bar{\tau}_1$ satisfy the same properties, we will use in the following the symbol $\bar{\tau}$ to refer to both of them. We stress that $\bar{\tau}$ is computationally cheaper than the endomorphism τ of the Koblitz curves discussed in the previous section. Indeed, the computational cost of $\bar{\tau}$ is of two squaring operations instead of the four squaring operations associated with τ. As it will be further discussed in Sect. 7, this computational saving induces an

$$\begin{tikzcd}
E_{(\gamma,1)} \arrow[d, "\tau_2"'] & & E_{(\gamma^2,1)} \arrow[d, "\tau_2"'] \\
E_{(\gamma^2,1)} \arrow[u, bend right=70, "\phi_0"'] & & E_{(\gamma,1)} \arrow[u, bend right=70, "\phi_1"']
\end{tikzcd}$$

Fig. 1. Construction of the endomorphisms $\bar{\tau}_0$ and $\bar{\tau}_1$ for the isogeny class-3 elliptic curves $E_{(\gamma,1)}$ and $E_{(\gamma^2,1)}$.

important reduction in the number of pre-computed points for the point multiplication $Q = kP$ that uses a width-w τNAF scalar representation.

Another interesting property of the $\bar{\tau}$ endomorphism is that $\bar{\tau}^2(x, y) = (x^4, y^4+x^4) = -\tau(x, y)$. Moreover, for the elliptic curves in isogeny class 3, τ satisfies the equation $\tau^2 + 4 = 3\tau$, which implies that $\bar{\tau}^2 + \bar{\tau} = -2$. It also follows that $\bar{\tau} = (-1 + \sqrt{-7})/2$. Since the ring $\mathbb{Z}[(-1 + \sqrt{-7})/2]$ has been extensively studied in the literature, we can adopt the same existing methods reported in [3,18,84,85] for performing the width-w τNAF scalar recoding.

We computed the cardinality of the elliptic curves belonging to isogeny class 3 defined over the field $\mathbb{F}_{4^m}$ with m a prime extension in the range [127, 191]. From this experiment, we found out that the only extension of cryptographic interest is $m = 163$. Indeed, for this extension field $\mathbb{F}_{4^{163}}$, the cardinality of the elliptic curves in class 3 has the following integer factorization:

$$\texttt{0x2} \cdot \texttt{0x28D} \cdot \texttt{0xC8B90A95C20EE5BBC91D671B0CEFED2EA\textbackslash}$$
$$\texttt{7901F5CEAAA522F37A4E0D020A19EBBDC1D0437C458139}.$$

The largest prime factor above has a size of approximately 316 bits. Hence, its associated security level is of around 158 bits. This curve is comfortably above the 128-bit security level (even considering the criterion that for a given field extension m, a binary curve offers ten bits less of security than the number $\lfloor \frac{m}{2} \rfloor$).

4. Base Field Arithmetic

In this section, we introduce practical aspects of the Koblitz curves previously presented. More particularly, we describe techniques for an efficient software implementation of 128-bit secure scalar multiplication algorithms over selected Koblitz curves defined over $\mathbb{F}_4$. Our library was specially designed for high-end 64-bit processors embedded with a 64-bit carry-less multiplication instruction and 128-bit vector registers that store and simultaneously process two 64-bit words [1,43].

We based our curve selection on two factors: security and performance. For conservative scenarios, we propose the class-3 curve over $\mathbb{F}_{4^{163}}$. This curve offers about 156 bits of security, which is well above the 10-bit security margin for binary curves (see Sect. 1 for further discussion). Moreover, it is equipped with a faster endomorphism when compared with class-1 and class-2 curves. For other scenarios, we suggest the class-1 curve over $\mathbb{F}_{4^{149}}$, which was chosen because of its 254-bit prime subgroup order, yielding a security level of approximately 128 bits.

4.1. *Modular Reduction*

One can construct a binary extension field $\mathbb{F}_{2^m}$ by taking a polynomial $f(x) \in \mathbb{F}_2[x]$ of degree m, which is irreducible over $\mathbb{F}_2$. It is very important that the form of the polynomial $f(x)$ admits an efficient modular reduction. The criteria for selecting $f(x)$ depend on the architecture to be implemented as it was extensively discussed in [78].

For our field extension degrees $m \in \{149, 163\}$, we do not have irreducible trinomials in the ring $\mathbb{F}_2[x]$. The immediate solution is to construct the fields $\mathbb{F}_{2^{149}}$ and $\mathbb{F}_{2^{163}}$ through irreducible pentanomials. However, the cost of a modular reduction with pentanomials is excessively high when compared with the field multiplication computed with carry-less instructions. This is because we need to perform four shift-and-add operations per reduction step. Besides, most of those operations require costly shift instructions, since the number of bits to be shifted is often not divisible by the word size (64 bits in our target architecture).

As a consequence, we resorted to the redundant trinomials strategy introduced in [21, 26]. Given a non-irreducible trinomial $g(x)$ of degree n that factorizes into an irreducible polynomial $f(x)$ of degree $m < n$, the idea is to perform the field reduction modulo $g(x)$ throughout the scalar multiplication and, at the end of the algorithm, reduce the polynomials modulo $f(x)$. Considering that our target platform counts with a native 64-bit carry-less multiplier, an efficient representation of the field elements must have at most 192 bits, that is, three 64-bit words. Furthermore, given a redundant trinomial $g(x) = x^n + x^a + x^b$, we need that most of the following constraints be satisfied for an efficient reduction:

(R1) The difference $(n - a) \geq 64$, which allows us to perform the reduction in a optimal number of steps in the interleaved representation (see Sect. 5 for more details).
(R2) The properties $(n - a) \equiv 0 \pmod{64}$, $(n - b) \equiv 0 \pmod{64}$ result in faster shift-and-add steps after the field multiplication and squaring operations. This is because they avoid additions between words, which require a series of left and right shifts to align the data within the registers.
(R3) The properties $(n - a) \equiv \pm 1 \pmod{64}$, $(n - b) \equiv \pm 1 \pmod{64}$ result in faster shift-and-add steps after the field squaring operation. The squaring operation consists of interleaving zeroes between the bits of the binary representation of a field element [40]. Therefore, parameters with this configuration help us avoiding shifts by one bit and one field addition in each reduction step.

For constructing the extension field $\mathbb{F}_{4^{149}}$, we selected the trinomial $g_{149}(x) = x^{192} + x^{19} + 1$, since it complies with the criteria R1 and R2. This polynomial factorizes into a 69-term irreducible polynomial $f_{149}(x)$ of degree 149. The field $\mathbb{F}_{4^{163}}$ is built via the trinomial $g_{163}(x) = x^{192} + x^3 + x^2$, which only satisfies the constraint R1. This result will affect the efficiency of the basic arithmetic operations; however, there were no better options for constructing this field. The trinomial $g_{163}(x)$ factorizes into an 87-term irreducible polynomial $f_{163}(x)$ of degree 163. The final reduction by $f_m(x)$ is performed via the mul-and-add approach[12] which, experimentally, performed more efficiently than the shift-and-add reduction for irreducible polynomials with large number of terms.

[12]For a more detailed explanation of the shift-and-add and the mul-and-add reduction methods for binary fields, see [18].

5. Quadratic Field Arithmetic

The quadratic field is constructed by the degree-two irreducible polynomial $h(u) = u^2 + u + 1$. Elements in this field are represented as $a_0 + a_1 u$, with $a_0, a_1 \in \mathbb{F}_{2^m}$. As discussed in Sect. 1, the $\mathbb{F}_{4^m}$ arithmetic can be implemented in terms of operations in the base field, resulting in considerable speedups due to the internal processor parallelism.

5.1. *Register Allocation*

The first question to be addressed on implementing an efficient quadratic field arithmetic is: How to allocate the binary representation of the field elements into the architecture available registers? In our case, we have to store two 192-bit polynomials into vector registers of 128 bits so that a fast modular reduction is possible, and the overhead in the two main arithmetic operations (i.e., multiplication and squaring) is kept to a minimum.

Let us consider an element $a = (a_0 + a_1 u) \in \mathbb{F}_{4^m}$, where $a_0 = Cx^{128} + Bx^{64} + A$ and $a_1 = Fx^{128} + Ex^{64} + D$ are 192-bit polynomials, each one of them stored into three 64-bit words (A-C, D-F). Also, suppose we have three 128-bit registers R_i, with $i \in \{0, 1, 2\}$, which can store two packed 64-bit words each.[13] The first option is to rearrange the extension field element a as

$$R_0 = A|B, \quad R_1 = C|D, \quad R_2 = E|F.$$

The problem with this representation is that a significant overhead is generated in the multiplication function, more specifically in the pre-computation phase of the Karatsuba procedure (cf. Sect. 5.2 with the computation of $V_{0,1}$, $V_{0,2}$, and $V_{1,2}$). Besides, in order to efficiently perform the subsequent reduction phase, we must temporarily store the polynomial terms into four 128-bit vectors, which could cause a register overflow. A better method for storing the element a is to use the interleaved arrangement as follows:

$$R_0 = D|A, \quad R_1 = E|B, \quad R_2 = F|C.$$

Using this setting, some overhead in the multiplication and squaring operations still exists, even though the penalty on the latter operation is almost negligible. In the positive side, the terms of the elements a_0, a_1 do not need to be rearranged and the modular reduction of these two base field elements can be performed in parallel.

5.2. *Multiplication*

Given two $\mathbb{F}_{4^m}$ elements $a = (a_0 + a_1 u)$ and $b = (b_0 + b_1 u)$, with a_0, a_1, b_0, b_1 in $\mathbb{F}_{2^m}$, we perform the multiplication $c = a \cdot b$ as

$$\begin{aligned} c = a \cdot b &= (a_0 + a_1 u) \cdot (b_0 + b_1 u) \\ &= (a_0 b_0 \oplus a_1 b_1) + (a_0 b_0 \oplus (a_0 \oplus a_1) \cdot (b_0 \oplus b_1))u, \end{aligned}$$

[13] In this document, we represent a 128-bit register R with its most (M) and least (L) significant packed 64-bit words as $R = M|L$.

where each element $a_i, b_i \in \mathbb{F}_{2^m}$ is composed of three 64-bit words. The analysis of the Karatsuba algorithm cost for different word sizes was presented in [88]. There, it was shown that the most efficient way to multiply three-word polynomials $s(x) = s_2x^2 + s_1x + s_0$ and $t(x) = t_2x^2 + t_1x + t_0$ as $v(x) = s(x) \cdot t(x)$ is through the one-level Karatsuba method:

$$V_0 = s_0 \cdot t_0, \quad V_1 = s_1 \cdot t_1, \quad V_2 = s_2 \cdot t_2,$$
$$V_{0,1} = (s_0 \oplus s_1) \cdot (t_0 \oplus t_1), \quad V_{0,2} = (s_0 \oplus s_2) \cdot (t_0 \oplus t_2) \quad V_{1,2} = (s_1 \oplus s_2) \cdot (t_1 \oplus t_2),$$
$$v(x) = V_2x^4 + (V_{1,2} \oplus V_1 \oplus V_2)x^3 + (V_{0,2} \oplus V_0 \oplus V_1 \oplus V_2)x^2 + (V_{0,1} \oplus V_0 \oplus V_1)x + V_0,$$

which costs six multiplications and 12 additions.

The algorithm requires six carry-less instructions, six vectorized `xor`s, and three bitwise shift instructions. In order to calculate the total multiplication cost, it is necessary to include the Karatsuba pre-computation operations at the base field level (12 vectorized `xor`s and six byte interleaving instructions) and at the quadratic field level (six vectorized `xor`s). Also, we must consider the reorganization of the registers in order to proceed with the modular reduction (six vectorized `xor`s).

5.3. *Modular Reduction*

The modular reduction of an element in $\mathbb{F}_{4^m}$ highly benefits from the interleaved representation, since the two packed words in a 128-bit register do not change their positions. In other words, the real (a_0) and the imaginary (a_1) parts are always placed in the least and most significant 64-bit words, respectively. The trinomial $g_{149}(x) = x^{192} + x^{19} + 1$ satisfies condition R2 (cf. Sect. 4.1) that is, $(192 - 0) \bmod 64 = 0$. For that reason, each step of the shift-and-add algorithm consists of only two shifts and three `xor`s, and three such steps are required.

Nonetheless, for the $\mathbb{F}_{4^{163}}$ case, we have to compute four shifts and four `xor`s per step. In the total cost, the modular reduction in this field requires three extra shifts and `xor`s when compared with the same operation over $\mathbb{F}_{4^{149}}$. Although this difference does not appear to be significant, its impact can be seen in the total cost of the scalar multiplication algorithm (see Sect. 7 for specific timings).

5.4. *Squaring*

Squaring is a very important function in the Koblitz curve point multiplication algorithm, since it is the building block for computing the τ endomorphism in both curve classes. In our implementation, we computed the squaring operation through carry-less multiplication instructions which, experimentally, was an approach less expensive than the bit interleaving method [40, §2.3.4]. The preprocessing phase is straightforward, and we just need to rearrange the 32-bit packed words of the 128-bit registers in order to prepare them for the subsequent modular reduction. The pre- and post-processing phases require three 32-bit shuffle instructions,[14] three vectorized `xor`s, and three bitwise shifts.

[14]On recent Intel architectures, this instruction is denominated `pshufd` [43].

5.5. *Inversion*

The inversion operation is computed via the Itoh–Tsujii method [44]. Given an element $c \in \mathbb{F}_{2^m}$, we compute $c^{-1} = c^{(2^{m-1}-1)\cdot 2}$ through an addition chain, where each step computes the term $(c^{2^i-1})^{2^j} \cdot c^{2^j-1}$ with $0 < j < i \leq m-1$. For the case $m = 149$, the following chain is used:

$$1 \to 2 \to 4 \to 8 \to 16 \to 32 \to 33 \to 66 \to 74 \to 148,$$

while for $m = 163$, we used

$$1 \to 2 \to 4 \to 5 \to 10 \to 20 \to 40 \to 80 \to 81 \to 162.$$

Both addition chains are optimal and were found through the procedure described in [22]. Note that although we computed the inversion operation over polynomials in $\mathbb{F}_2[x]$ (reduced modulo $g_m(x)$), we still had to build the addition chain with $m \in \{149, 163\}$, since we are in fact interested in the embedded $\mathbb{F}_{2^m}$ field element.

6. Scalar Multiplication Algorithms

In this part, we briefly discuss single-core algorithms for computing a timing-resistant scalar multiplication function over Koblitz curves. The fastest approach is to compute the function via the left-to-right or right-to-left τ-and-add algorithms [40]. A more conservative and simpler method is to perform the point multiplication with Montgomery ladders [63]. This method disregards the efficient τ endomorphism but takes profit of the special curve b-parameter within the x-coordinate doubling and addition formulas.

6.1. *Left-to-Right τ-and-Add*

This algorithm is similar to the traditional left-to-right double-and-add method. Here, the point doubling operation is replaced by the computationally cheaper τ endomorphism.[15] Before that, we need to compute the width-w τNAF representation of the scalar k. In addition, linear passes must be applied in the accumulators in order to avoid cache-based timing attacks (for more details, see Sect. 6.4). The process is shown in Algorithm 1.

The main advantage of this method is that the sensitive data are indirectly related to the points P_{v_i}, which are only read and then added to the unique accumulator Q. As a consequence, just one linear pass per iteration is required, immediately prior to reading P_{v_i}. On the other hand, the operation $\tau^{w-1}(Q)$ must be performed on three coordinates of a projective point by successive squarings, since computing it through lookup tables could leak information about the scalar k bits.[16]

[15]For the sake of simplicity, we will not differentiate the endomorphisms τ and $\bar{\tau}$ when describing the τ-and-add algorithms.

[16]Except for aggressive choices for the parameter w (usually in the fixed-point scenario), computing the τ endomorphism with lookup tables is not worthwhile in modern platforms. This is because performing the field

Algorithm 1 Left-to-right width-w τ-and-add on Koblitz curves over $\mathbb{F}_4$

```
Input: Koblitz curve $E/\mathbb{F}_{4^m}$, point $P \in E(\mathbb{F}_{4^m})$, $k \in \mathbb{Z}[\tau]$
Output: $Q = kP$
1: Compute the regular width-$w$ $\tau$NAF of $k$ as $\sum_{i=0}^{l} v_i \tau^{i(w-1)}$
2: Pre-compute the $2^{w-1}$ ($2^{2(w-1)-1}$) points $P_i \leftarrow \alpha_i P$ for class-3 (class-1,2) $E$
3: $Q \leftarrow \mathcal{O}$
4: for $i = l$ to 0 do
5:     $Q \leftarrow \tau^{w-1}(Q)$
6:     Perform a linear pass to recover $P_{v_i}$
7:     $Q \leftarrow Q \pm P_{v_i}$
8: end for
9: return $Q = kP$
```

Note that, in the context of the Diffie–Hellman key exchange protocol, we can assume that the scalar k is kept in the $\mathbb{Z}[\tau]$ form, since it is a private key and therefore (ideally) it is not manipulated by other entities other than its owner. This procedure, besides simplifying the implementation of the τ-and-add point multiplication, reduces the possibilities of timing attacks, given that Solinas' partial reduction algorithm contains branching sections [82]. If desired, the scalar can be retrieved as an integer in $\mathbb{Z}_r$ as discussed in Sect. 2.1.

6.2. *Right-to-Left τ-and-Add*

An alternative is to process the scalar k from the least to the most significant digit. Again, taking advantage of the τ endomorphism, the GLV method is brought to its full extent. This approach is called *right-to-left τ-and-add*, and it is presented in Algorithm 2.

Algorithm 2 Right-to-left τ-and-add on Koblitz curves over $\mathbb{F}_4$

```
Input: Koblitz curve $E/\mathbb{F}_{4^m}$, point $P \in E(\mathbb{F}_{4^m})$, $k \in \mathbb{Z}[\tau]$
Output: $Q = kP$
1: Compute the regular width-$w$ $\tau$NAF of $k$ as $\sum_{i=0}^{l} v_i \tau^{i(w-1)}$
2: Initialize the $2^{w-1}$ ($2^{2(w-1)-1}$) accumulators $Q_i \leftarrow \mathcal{O}$ for class-3 (class-1,2) $E$
3: for $i = 0$ to $l$ do
4:     Perform a linear pass to recover $Q_{v_i}$
5:     $Q_{v_i} \leftarrow Q_{v_i} \pm P$
6:     Perform a linear pass to store $Q_{v_i}$
7:     $P \leftarrow \tau^{w-1}(P)$
8: end for
9: $Q \leftarrow \mathcal{O}$
10: Compute the $2^{w-1}$ ($2^{2(w-1)-1}$) points $Q \leftarrow \sum \alpha_i Q_i$ for class-3 (class-1,2) $E$
11: return $Q = kP$
```

Footnote 16 continued
squaring through carry-less instructions costs approximately five times the cost of computing a multi-squaring operation via pre-computed values.

Here, we have to perform a post-computation in the accumulators instead of pre-computing the points P_i as in the previous approach. Also, the τ endomorphism is applied to the point P, which is usually public. For that reason, we can compute τ with table lookups instead of performing squarings multiple times. The downside of this algorithm is that the accumulators carry sensitive information about the private key. Moreover, they are read and written. As a result, it is necessary to apply the linear pass algorithm over the whole set of accumulators Q_i twice per loop iteration.

6.3. *Montgomery Ladder*

The scalar multiplication can also be computed via Montgomery ladders, which provide simple and efficient timing-resistant formulas. The downside of this approach is that the speedup from the Frobenius map is lost. On the other hand, the Montgomery formulas are significantly improved when applied with the Koblitz curve b-parameter. For instance, the doubling formula only requires three field squarings and one multiplication. Moreover, if we choose the group generator with the x-coordinate equal to $x^{64}u$, the cost of one loop multiplication is reduced to 16 logical vector instructions. For fixed-point scenarios, the right-to-left Montgomery ladder with Koblitz curves efficiently computes the point multiplication with $\log_2(r)$ pre-computed x-coordinates. (see [72] for more details on the Montgomery formulas for binary curves). The advantage of using this method is that no linear passes are required, when compared with windowed versions of the τ-and-add algorithm.

For Montgomery ladders, we suggest constructing the scalar k as follows. First, 40 bytes K_i are chosen at random from [0, 255]. Then, we update the most and least significant byte as

$$K_{39} \leftarrow 0x5, \quad K_0 \leftarrow K_0 \vee 0x1.$$

Next, the scalar is multiplied by 653, which is one of the factors of $\#E(\mathbb{F}_{4^{163}})$. Finally, after processing the scalar multiplication, the accumulator was doubled in order to return $Q = 2kP$. By doing this, we assured that the least and most significant bit of k is always equal to one, avoiding any inconsistencies in the Montgomery addition left-to-right and right-to-left binary formulas.

Remark on ECDSA applications The lattice-based attacks presented in [7,68] show that fixing bits of the scalar k in the signing phase of the ECDSA potentially reveals the secret key after the collection of a few hundred signatures. Nguyen and Shparlinski [68] describe how to reduce the ECDSA problem to the Hidden Number Problem (HNP) if the ℓ most (or least) significant bits of the scalar are known. Here, $\ell = \lceil \log \log r \rceil$, where r is the order of the subgroup of interest. For our first curve, r is a 316-bit prime. Hence, $\ell = 9$ consecutive biased bits would be necessary in order to launch the ECDSA-HNP attack. Since in our setting we are only fixing three bits (the most and the two least significant bits), then it would appear that our configuration is safe against this kind of attacks.

Nevertheless, as demonstrated in [7], a concrete analysis of the attack could improve the aforementioned bound. Moreover, since the effects that this type of attacks may have on points of composite order are not clearly understood, we do not recommend the usage of Montgomery ladders in the ECDSA signing phase. Instead, the designer may use τ-and-add/double-and-add methods, which do not fix any bit of the scalar k. In cases where the implementer is compelled to use Montgomery ladders, the scalar should be selected at random from $[1, r-1]$. As a consequence, the order of the input point has to be verified. Finally, to avoid formula inconsistencies, the order r must be added to k if the nonce is an even number.

6.4. *Auxiliary Functions*

In the next paragraphs, we comment on the implementation aspects of functions that support the aforementioned scalar multiplication algorithms.

6.4.1. *Regular τNAF Representation*

The regular NAF representation was proposed by Joye [48] and adopted for the $\mathbb{Z}[\tau]$ ring in [69]. This approach is crucial for implementing timing-resistant NAF-based scalar multiplication algorithms on curves with non-complete addition formulas. The general procedure is presented in Algorithm 3.

Algorithm 3 Regular width-w τNAF recoding

Input: t_w [81, §7.3], $\alpha_u = \beta_u + \gamma_u \tau$ (see Table 2), $k = r_0 + r_1\tau \in \mathbb{Z}[\tau]$ with odd r_0, r_1
Output: $\rho = \sum_{i=0}^{l} v_i \tau^{i(w-1)}$

```
1: for $i \leftarrow 0$ to $l-1$ do
2:     if $w = 2$ then
3:         $v_i \leftarrow ((r_0 - 4 \cdot r_1) \bmod 8) - 4$
4:         $r_0 \leftarrow r_0 - v_i$
5:     else
6:         $u \leftarrow (r_0 + r_1 t_w \bmod 2^{2w-1}) - 2^{2(w-1)}$
7:         if $u > 0$ then $s \leftarrow 1$ else $s \leftarrow -1$
8:         $r_0 \leftarrow r_0 - s\beta_u, r_1 \leftarrow r_1 - s\gamma_u, v_i \leftarrow s\alpha_u$
9:     end if
10:    for $j \leftarrow 0$ to $(w-2)$ do
11:        $t \leftarrow r_0, r_0 \leftarrow r_1 + (\mu \cdot r_0)/4, r_1 \leftarrow -t/4$
12:    end for
13: end for
```

The operations throughout the algorithm are performed via basic arithmetic and logic machine operations, with the subscalars r_0, r_1 represented in the two's complement form. As a result, the negation can be implemented through a masking function. Each of the β_u and γ_u values and their respective signs are stored into a single 64-bit register, which is accessed via bitwise shift and logical operators. Finally, to avoid carry-related branches, the multiword addition is programmed in Assembly code with the addition-with-carry instructions (`addq`, `adcq`) [43].

6.4.2. *Linear Pass*

The linear pass is a method designed to protect sensitive information against side-channel attacks associated with the CPU cache access patterns [73,86]. Let us consider an array A of size l. Before reading a value $A[i]$, with $i \in \{0, \dots, l-1\}$, the linear pass technique reads the entire array A but only stores, usually into an output register, the requested data $A[i]$.

As a consequence, the attacker does not know which array index was accessed just by analyzing the location of the cache miss in his own artificially injected data. Writing in $A[i]$ occurs in a similar vein. Assuming that the data to be written are stored into a register, we proceed by deceitfully updating all values of A, except for $A[i]$, which receives the real data. Note that this method causes a considerable overhead directly related to the size of the array.

In this work, we implemented the linear pass method using 128-bit SSE vectorized instructions and registers. The selection of $A[i]$ is performed via logical functions and the SSE instruction `pcmpeqq` that compares the values of two 128-bit registers A and B and sets the resulting register C with bits one, if A and B are equal, and bits zero otherwise [43].

6.4.3. *Conditional Swap*

The conditional swap is a technique usually employed in Montgomery ladders to hide the access order of the two accumulators $R_0 and R_1$. The knowledge of this order by a malicious entity could result in catastrophic consequences for the cryptosystem [37]. The countermeasure can be similarly applied to width-w versions of τ-and-add algorithms that process only two accumulators. As a result, the double linear pass in the right-to-left variant is not required anymore. For τ-and-add algorithms where more than two accumulators are processed, it is possible to extend the conditional swap to n points in $log_2(n)$ steps. However, the required number of `xor` instructions results in a greater overhead than the double linear pass discussed above. Finally, we can perform the conditional swap only in the lowest 2^{n-1} array A values. Nonetheless, this approach requires the storage of the array state.

7. Results and Discussion

Our software library was designed for 64-bit high-end desktops, provided with SSE 4.1-equivalent vector instructions and a 64-bit carry-less multiplier. The timings were measured in an Intel Core i7 4770k 3.50 GHz machine (Haswell architecture) with the Turbo Boost and Hyper-Threading technologies disabled. The implementation was coded in the GNU11 C and Assembly languages.

We compiled our code with the GCC (Gnu Compiler Collection) version 5.4 with the optimization flags `--march=haswell -fomit-frame-pointer -O3`. In addition, the 3-τNAF left-to-right τ-and-add point multiplication over $E_1/F_{4^{149}}$ code is available in the eBACS platform [9].

Table 4. Timings for the finite field arithmetic in $\mathbb{F}_{4^{149}}$ and $\mathbb{F}_{4^{163}}$.

Field operation	$\mathbb{F}_{4^{149}}$		$\mathbb{F}_{4^{163}}$	
	Cost (cc)	Ratio op/m	Cost (cc)	Ratio op/m
General multiplication (m)	52	1.00	68	1.00
Multiplication by $x^{64}u$	–	–	12	0.18
Squaring	20	0.38	28	0.41
Inversion	6600	126.92	6300	92.65
Reduction modulo $f_m(x)$	452	8.69	432	6.35

Remark on error margin The analysis in [74] shows that our machine has an error margin of four clock cycles. This value is not significant when considering the point arithmetic or scalar multiplication timings. Still, for simpler functions such as the basic finite field operations, it is recommended to consider this remark.

7.1. *Parameters*

Our base binary fields $\mathbb{F}_{2^m} \cong \mathbb{F}_2[x]/(f_m(x))$ with $m \in \{149, 163\}$ were constructed by the 69- and 87-term irreducible polynomials $f_{\{149,163\}}(x)$ described in Sect. 5. The quadratic extension $\mathbb{F}_{q^2} \cong \mathbb{F}_q[u]/(h(u))$ was built through the irreducible quadratic $h(u) = u^2 + u + 1$.

Our class-1 Koblitz curve is defined as $E_{(0,u)}/\mathbb{F}_{4^{149}} : y^2 + xy = x^3 + ux^2 + u$, and the group $E_{(0,u)}(\mathbb{F}_{4^{149}})$ contains a subgroup of interest of order

$$r = \texttt{0x637845F7F8BFAB325B85412FB54061F148B7F6E79AE11CC843ADE1470F7E4E29},$$

which corresponds to approximately 254 bits. Our class-3 curve is defined as $E_{(u,1)}/\mathbb{F}_{4^{163}} : y^2 + xy = x^3 + ux^2 + 1$, and the order of the prime subgroup of $E_{(u,1)}(\mathbb{F}_{4^{163}})$ is

$$r = \texttt{0xC8B90A95C20EE5BBC91D671B0CEFED2EA701F5CE\textbackslash}$$
$$\texttt{9AAA522F37A4E0D020A19EBBDC1D0437C458139},$$

of about 316 bits. In addition, throughout our τ-and-add scalar multiplication implementation, we represented the points in λ-affine [50,77] and λ-projective [71] coordinates.

7.2. *Field and Elliptic Curve Arithmetic Timings*

We present in Table 4 the timings for the quadratic field arithmetic. The field multiplication, squaring, and inversion timings already include the costs for the modular reduction modulo $g_m(x)$. Because of our machine error margin, we abstained from presenting timings for the field addition (implemented by three `xor` instructions), which theoretically costs from 1.66 to three clock cycles. The column *Ratio op*/m shows the ratio between the timings of the presented operation and the field multiplication.

Table 5. Timings for the point arithmetic in $E_{(0,u)}(\mathbb{F}_{4^{149}})$ and $E_{(u,1)}(\mathbb{F}_{4^{163}})$.

Point operation	$E_{(0,u)}(\mathbb{F}_{4^{149}})$		$E_{(u,1)}(\mathbb{F}_{4^{163}})$	
	Cost (cc)	Ratio op/m	Cost (cc)	Ratio op/m
λ-Doubling	368	7.08	420	6.18
λ-Full addition	792	15.23	828	12.18
λ-Mixed addition	592	11.38	624	9.18
Montgomery doubling	–	–	168	2.47
Montgomery left-to-right addition	–	–	272	4.00
Montgomery right-to-left addition	–	–	388	5.71
Affine Frobenius map	80 (τ)	1.54	64 ($\bar{\tau}$)	0.94
Projective Frobenius map	120 (τ)	2.31	108 ($\bar{\tau}$)	1.59

The ratio squaring/multiplication is relatively high when compared with other binary curves such as GLS-254 [71]. This is because here the trinomials $g_m(x)$ do not admit a reduction specially crafted for the squaring operation (see Sect. 4.1). Table 4 also shows that the field inversion is significantly more expensive than a multiplication and therefore, should be avoided as much as possible in our scalar multiplication design. The high cost of this operation is explained by the fact that, to avoid attacks on projective coordinates [64], we did not work with lookup tables for computing the different Itoh–Tsujii multi-squaring operations, but applied consecutive field squarings. Finally, the result of having a redundant trinomial which does not comply with criteria R2 (see Sect. 4.1) is indicated above: sixteen and eight extra cycles for the field multiplication and squaring, respectively. Next, the timings for the point arithmetic used in our implementation are shown in Table 5.

The effects of a slower field arithmetic is seen in Table 5. Over $E_{(u,1)}(\mathbb{F}_{4^{163}})$, the λ-doubling, -full addition, and -mixed addition[17] are 1.14, 1.05, and 1.05 slower when compared with the same operations over $E_{(0,u)}(\mathbb{F}_{4^{149}})$. In the other hand, the faster class-3 endomorphisms outperform the class-1 equivalents by 20 and 10% in the affine and projective formulas, respectively.

7.3. *Scalar Multiplication Timings*

At last, we present in Table 6 the results obtained for the scalar multiplication implementation through the τ-and-add (with different values for the width w) and Montgomery ladder algorithms. After calculating estimations, we saw that, over the $E_{(0,u)}/\mathbb{F}_{4^{149}}$ curve, the pre- and post-computation costs along with the timing-attack countermeasures overhead would surpass the savings in the total number of point operations. In the class-3 curve $E_{(u,1)}/\mathbb{F}_{4^{163}}$, this scenario occurs when $w > 5$. Note that the same parameter w represents a different number of pre-/post-computed points when comparing the two curve classes, since the $\tau, \bar{\tau}$ endomorphisms and the regular recoding algorithms are different.

For the sake of comparison, Table 6 includes the ratio clock cycles per bit. This is because it is inaccurate to compare the absolute timings of each curve, since the

[17]In the τ-and-add algorithms, the λ-doubling and the λ-full addition formulas are only used in the pre- and post-computation phases.

Table 6. Timings for the τ-and-add scalar multiplication on $E_{(0,u)}/\mathbb{F}_{4^{149}}$ and $E_{(u,1)}/\mathbb{F}_{4^{163}}$.

	$E_{(0,u)}/\mathbb{F}_{4^{149}}$		$E_{(u,1)}/\mathbb{F}_{4^{163}}$	
	Cost (cc)	Ratio cycles (bit)	Cost (cc)	Ratio cycles (bit)
Left-to-right τ-and-add				
$w = 2$	111,420	442.14	234,880	752.56
$w = 3$	82,872	328.85	144,988	464.70
$w = 4$	100,692	399.57	116,560	373.58
$w = 5$	–	–	115,420	369.93
Right-to-left τ-and-add				
$w = 2$	105,164	417.31	221,132	708.75
$w = 3$	85,404	338.90	128,980	413.39
$w = 4$	141,456	561.30	105,952	339.58
$w = 5$	–	–	116,888	374.64

Table 7. Fraction (in percent) of the linear pass countermeasure on the cost of different τ-and-add algorithms over $E_{(0,u)}/\mathbb{F}_{4^{149}}$.

Left-to-right			Right-to-left		
$w = 2$	$w = 3$	$w = 4$	$w = 2$	$w = 3$	$w = 4$
2.10	2.76	10.90	8.82	18.76	36.57

security provided by the class-3 curve is higher than our proposed class-1 curve. The ratio is calculated over the number of randomly chosen bits processed by the τ-and-add algorithms. In the curve $E_{(0,u)}/\mathbb{F}_{4^{149}}$ implementation, 252 bits are considered, while in the $E_{(u,1)}/\mathbb{F}_{4^{163}}$ design, we have a total of 312 bits.

The above results show that for the class-1 case, the costs of the pre- and post-computation increase rapidly with the parameter w. The difference in the point pre-computation timings between widths $w = 2$ and 3 is about 86.93%, while between widths $w = 3$ and 4, the variation is approximately 168.97%. This is because the number of points to be computed quadruples at each value w. This aspect is also reflected in the cost of the linear pass functions, which also depends on the number of computed points. Table 7 describes the fraction of the scalar multiplication cost dedicated to applying the linear pass function in different τ-and-add approaches.

The same issue does not occur in the class-3 curve, since the number of points at each increasing w-value only doubles. The bottleneck here lies in the base field arithmetic, as discussed in the previous paragraphs. Nevertheless, the class-3 curve obtained competitive timings when compared with its counterpart. In terms of cost per bit, its best algorithm, namely the right-to-left variant with $w = 4$, is only 1.03 times more expensive than the left-to-right version with $w = 3$ over our class-1 curve. This result is clearly due to the faster τ endomorphisms and the smaller amount of points to protect in the main loop. At last, we present in Table 8 the timings for the Montgomery ladder algorithms.

The savings in the left-to-right Montgomery addition formulas resulted in competitive timings when compared with the τ-and-add algorithms with $w \leq 3$. The advantage of the

Table 8. Timings for the Montgomery ladder scalar multiplication $E_{(u,1)}/\mathbb{F}_{4^{163}}$.

Algorithm	$E_{(u,1)}/\mathbb{F}_{4^{163}}$	
	Cost (cc)	Ratio cycles (bit)
Left-to-right	145,188	465.34
Right-to-left (with pre-computation)	128,284	411.16

Table 9. A comparison between state-of-the-art software implementations of 128-bit secure timing-resistant scalar multiplication.

Curve/method	Cost (cc)
Koblitz over $\mathbb{F}_{2^{283}}$ (τ-and-add, 5-τNAF) [69]	99,000
GLS over $\mathbb{F}_{4^{127}}$ (2-GLV double-and-add, 5-NAF) [70]	48,300
Twisted Edwards over $\mathbb{F}_{(2^{127}-1)^2}$ (double-and-add) [24]	56,000
Kummer genus-2 over $\mathbb{F}_{2^{127}-1}$ (Kummer ladder) [8]	60,556
Koblitz over $\mathbb{F}_{4^{149}}$ (τ-and-add, 3-τNAF) (this work)	82,872
Koblitz over $\mathbb{F}_{4^{163}}$ (τ-and-add, 4-τNAF) (this work)	105,952

The timings were measured in the Haswell platform and are given in clock cycles

Montgomery ladder is that it is simpler to implement and does not require complicated countermeasures for protecting the multiple pre-computed points and accumulators. The right-to-left Montgomery ladder algorithm results do not compare well with the costs presented in Table 6, since this approach is applied in a different setup, namely the fixed-point scenario.

7.4. *Comparison*

At last, we compare our best point multiplication versions against the state-of-the-art 128-bit secure timing-resistant software scalar multiplication implementations on binary and prime curves. The results are presented in Table 9.

The 3-τNAF scalar multiplication on $E_{(0,u)}/\mathbb{F}_{4^{149}}$ is the fastest published software implementation on 128-bit secure Koblitz curves, surpassing the work on the standardized curve K-283 [69] by 20.29%. On binary fields, the GLS implementation in [70] outperforms our faster implementation by 38.80%. The works in [24] and [8] are about 29.04 and 23.27% faster, respectively. Finally, our $E_{(u,1)}/\mathbb{F}_{4^{163}}$ curve is only 6000 cycles more expensive than the K-283 implementation in [69], but offers 15 extra bits of security. Given that both implementations work with the same endomorphism τ, we can see that the quadratic extension plays an important role in the efficiency of the field arithmetic.

8. Conclusion

In this work, we gave a comprehensive taxonomy of Koblitz-like curves defined over the extension fields $\mathbb{F}_{4^m}$ with good cryptographic properties. We also introduced a family of curves, whose rich structure admits novel and more efficient endomorphisms than the traditional Frobenius endomorphism associated with classical Koblitz curves.

We presented fast software implementations of 128-bit secure scalar multiplication algorithms on the Koblitz curves $E_{(0,u)}/\mathbb{F}_{4^{149}}$ and the Koblitz-like curve $E_{(1,u)}/\mathbb{F}_{4^{163}}$. The design of these implementations included original techniques to construct the basic finite field arithmetic and the methods to protect the code against timing attacks. An important feature that distinguishes our code from previous binary field arithmetic implementations is the intensive usage of the carry-less multiplication instruction. The drastic reduction of its latency and throughput in the recent architectures enabled its application to operations that go beyond the field multiplication, such as the field squaring and modular reduction.

Acknowledgements

We thank Diego Aranha for pointing out the work of Naccache, Smart, and Stern on sensitive information leak related to projective coordinates.

References

[1] AMD Technology, AMD64 architecture programmer's manual, Volume 1: Application programming. 24592 3.21. http://developer.amd.com/resources/developer-guides-manuals/

[2] ANSSI, Les Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques. Agence nationale de la sécurit des systèmes dinformation (2014). https://www.ssi.gouv.fr/guide/cryptographie-les-regles-du-rgs/

[3] D.F. Aranha, A. Faz-Hernández, J. López, F. Rodríguez-Henríquez, Faster implementation of scalar multiplication on Koblitz curves, in *Proceedings of LATINCRYPT 2012*. LNCS, vol. 7533 (Springer, Berlin, 2012), pp. 177–193

[4] D.F. Aranha, J.López, D. Hankerson, Efficient software implementation of binary field arithmetic using vector instruction sets, in *Proceedings of LATINCRYPT 2010*. LNCS, vol. 6212 (Springer, Berlin, 2010), pp. 144–161

[5] A.U. Ay, E. Öztürk, F. Rodríguez-Henríquez, E. Savaş, Design and implementation of a constant-time FPGA accelerator for fast elliptic curve cryptography, in *ReConFig 2016* (IEEE, Piscataway, 2016), pp. 1–8

[6] R. Barbulescu, P. Gaudry, A. Joux, E. Thomé, A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, in *Proceedings of EUROCRYPT 2014*. LNCS, vol. 8441 (Springer, Berlin, 2014), pp. 1–16

[7] P. Belgarric, P.-A. Fouque, G. Macario-Rat, M. Tibouchi, Side-channel analysis of Weierstrass and Koblitz curve ECDSA on Android smartphones, in *Proceedings of CT-RSA 2016*. LNCS, vol. 9610 (Springer, Berlin, 2016), pp. 236–252

[8] D.J. Bernstein, C. Chuengsatiansup, T. Lange, P. Schwabe, Kummer strikes back: new DH speed records, in *Proceedings of ASIACRYPT 2014*. LNCS, vol. 8873 (Springer, Berlin, 2014), pp. 317–337

[9] D.J. Bernstein, T.L. (eds.), eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to. Accessed 14 Dec 2016

[10] D.J. Bernstein, S. Engels, T. Lange, R. Niederhagen, C. Paar, P. Schwabe, R. Zimmermann, Faster discrete logarithms on FPGAs. Cryptology ePrint Archive, Report 2016/382 (2016). http://eprint.iacr.org/2016/382

[11] D.J. Bernstein, T. Lange, eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to. Accessed 12 Dec 2016

[12] D.J. Bernstein, T. Lange, SafeCurves: choosing safe curves for elliptic-curve cryptography. http://safecurves.cr.yp.to. Accessed 14 Dec 2016

[13] 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 *Proceedings of Pairing 2008*. LNCS, vol. 5209 (Springer, Berlin, 2008), pp. 297–315

[14] J. Beuchat, J. Detrey, N. Estibals, E. Okamoto, F. Rodríguez-Henríquez, Fast architectures for the η_T pairing over small-characteristic supersingular elliptic curves. *IEEE Trans. Comput.* **60**(2), 266–281 (2011)

[15] J. Beuchat, E. López-Trejo, L. Martínez-Ramos, S. Mitsunari, F. Rodríguez-Henríquez, Multi-core implementation of the Tate pairing over supersingular elliptic curves, in *Proceedings of CANS 2009*. LNCS, vol. 5888 (Springer, Berlin, 2009), pp. 413–432

[16] I.F. Blake, R. Fuji-Hara, R.C. Mullin, S.A. Vanstone, Computing logarithms in finite fields of characteristic two. *SIAM J. Algebr. Discrete Methods* **5**, 276–285 (1984)

[17] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, B. Moeller, Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS). RFC 4492. Internet Engineering Task Force (IETF) (2006). https://tools.ietf.org/html/rfc4492

[18] M. Bluhm, S. Gueron, Fast software implementation of binary elliptic curve cryptography. *J. Cryptogr. Eng.* **5**(3), 215–226 (2015)

[19] D. Boneh, M.K. Franklin, Identity-based encryption from the Weil pairing, in *Proceedings of CRYPTO 2001*. LNCS, vol. 2139 (Springer, Berlin, 2001), pp. 213–229

[20] J.W. Bos, C. Costello, P. Longa, M. Naehrig, Selecting elliptic curves for cryptography: an efficiency and security analysis. *J. Cryptogr. Eng.* **6**(4), 259–286 (2016)

[21] R.P. Brent, P. Zimmermann, Algorithms for finding almost irreducible and almost primitive trinomials, in *Primes and Misdemeanours: Lectures in Honour of the Sixtieth Birthday of Hugh Cowie Williams* (Fields Institute, Toronto, 2003), p. 212

[22] N.M. Clift, Calculating optimal addition chains. *Computing* **91**(3), 265–284 (2011)

[23] D. Coppersmith, Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Inf. Theory* **30**(4), 587–593 (1984)

[24] C. Costello, P. Longa, Four($\mathbb{Q}$): four-dimensional decompositions on a ($\mathbb{Q}$)-curve over the Mersenne prime, in *Proceedings of ASIACRYPT 2015*. LNCS, vol. 9452 (Springer, Berlin, 2015), pp. 214–235

[25] T. Dierks, E. Rescorla, The transport layer security (TLS) protocol version 1.2. RFC 5246. Internet Engineering Task Force (IETF) (2008). https://tools.ietf.org/html/rfc5246

[26] C. Doche, Redundant trinomials for finite fields of characteristic 2, in *Proceedings of ACISP 2005*. LNCS, vol. 3574 (Springer, Berlin, 2005), pp. 122–133

[27] ECRYPT II, Ecrypt II yearly report on algorithms and keysizes (2011–2012). Katholieke Universiteit Leuven (KUL) (2012). http://www.ecrypt.eu.org/

[28] A. Enge, P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arith.* **102**, 83–103 (2002)

[29] J. Faugère, L. Perret, C. Petit, G. Renault. Improving the complexity of index calculus algorithms in elliptic curves over binary fields, in *Proceedings of EUROCRYPT 2012*. LNCS, vol. 7237 (Springer, Berlin 2012), pp. 27–44

[30] S.D. Galbraith, P. Gaudry, Recent progress on the elliptic curve discrete logarithm problem. *Des. Codes Cryptogr.* **78**(1), 51–72 (2016)

[31] S.D. Galbraith, S.W. Gebregiyorgis, Summation polynomial algorithms for elliptic curves in characteristic two, in *Proceedings of INDOCRYPT 2014*. LNCS, vol. 8885 (Springer, Berlin, 2014), pp. 409–427

[32] S.D. Galbraith, X. Lin, M. Scott, Endomorphisms for faster elliptic curve cryptography on a large class of curves, in *Proceedings of EUROCRYPT 2009*. LNCS, vol. 5479 (Springer, Berlin, 2009), pp. 518–535

[33] S.D. Galbraith, N.P. Smart, A cryptographic application of Weil descent, in *Proceedings of Cryptography and Coding*. LNCS, vol. 1746 (Springer, Berlin, 1999), pp. 191–200

[34] R.P. Gallant, R.J. Lambert, S.A. Vanstone, Improving the parallelized pollard lambda search on anomalous binary curves. *Math. Comput.* **69**(232), 1699–1705 (2000)

[35] P. Gaudry, Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symb. Comput.* **44**(12), 1690–1702 (2009)

[36] P. Gaudry, F. Hess, N.P. Smart, Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptol.* **15**, 19–46 (2002)

[37] D. Genkin, L. Valenta, Y. Yarom, May the fourth be with you: a microarchitectural side channel attack on several real-world applications of curve25519. Cryptology ePrint Archive, Report 2017/806 (2017). https://eprint.iacr.org/2017/806

[38] R. Granger, T. Kleinjung, J. Zumbrägel, On the powers of 2. Cryptology ePrint Archive, Report 2014/300 (2014). http://eprint.iacr.org/2014/300

[39] D. Hankerson, K. Karabina, A. Menezes, Analyzing the Galbraith–Lin–Scott point multiplication method for elliptic curves over binary fields. *IEEE Trans. Comput.* **58**(10), 1411–1420 (2009)
[40] D. Hankerson, A.J. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography* (Springer, Secaucus, 2003)
[41] F. Hess, Generalising the GHS attack on the elliptic curve discrete logarithm problem. *LMS J. Comput. Math.* **7**, 167–192 (2004)
[42] Y.-J. Huang, C. Petit, N. Shinohara, T. Takagi, On generalized first fall degree assumptions. Cryptology ePrint Archive, Report 2015/358 (2015). http://eprint.iacr.org/2015/358
[43] Intel Corporation, Intel 64 and IA-32 architectures software developers manual, 253665-064US (2017)
[44] T. Itoh, S. Tsujii, A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. *Inf. Comput.* **78**(3), 171–177 (1988)
[45] A. Joux, A one round protocol for tripartite Diffie–Hellman, in *Proceedings of ANTS-IV*. LNCS, vol. 1838 (Springer, Berlin, 2000), pp. 385–394
[46] A. Joux, A one round protocol for tripartite Diffie–Hellman. *J. Cryptol.* **17**(4), 263–276 (2004)
[47] A. Joux, A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic, in *Proceedings of SAC 2013*. LNCS, vol. 8282 (Springer, Berlin, 2014), pp. 355–379
[48] M. Joye, M. Tunstall, Exponent recoding and regular exponentiation algorithms, in *AFRICACRYPT 2009*. LNCS, vol. 5580 (Springer, Berlin, 2009), pp. 334–349
[49] K. Karabina, Point decomposition problem in binary elliptic curves. Cryptology ePrint Archive, Report 2015/319 (2015). http://eprint.iacr.org/2015/319
[50] E. Knudsen, Elliptic scalar multiplication using point halving, in *Proceedings of ASIACRYPT 99*. LNCS, vol. 1716 (Springer, Berlin, 1999), pp. 135–149
[51] N. Koblitz, Elliptic curve cryptosystems. *Math. Comput.* **48**, 203–9 (1987)
[52] N. Koblitz, Constructing elliptic curve cryptosystems in characteristic 2, in *Proceedings of CRYPTO 90*. LNCS, vol. 537 (1990), pp. 156–167
[53] N. Koblitz, CM-curves with good cryptographic properties, in *Proceedings of CRYPTO 1991*. LNCS, vol. 576 (Springer, Berlin, 1991), pp. 279–287
[54] N. Koblitz, A. Menezes, A riddle wrapped in an enigma. Cryptology ePrint Archive, Report 2015/1018 (2015) http://eprint.iacr.org/2015/1018
[55] P.C. Kocher, J. Jaffe, B. Jun, Differential power analysis, in *Proceedings of CRYPTO 99*. LNCS, vol. 1666 (Springer, Berlin, 1999), pp. 388–397
[56] P. Longa, F. Sica, Four-dimensional Gallant–Lambert–Vanstone scalar multiplication. *J. Cryptol.* **27**(2), 248–283 (2014)
[57] M. Maurer, A. Menezes, E. Teske, Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree, in *Proceedings of INDOCRYPT 2001*. LNCS, vol. 2247 (Springer, Berlin, 2001), pp. 195–213
[58] A. Menezes, T. Okamoto, S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inf. Theory* **39**, 1639–1646 (1993)
[59] A. Menezes, T. Okamoto, S.A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, in *STOC 91* (ACM, New York, 1992), pp. 80–89
[60] A. Menezes, M. Qu, Analysis of the Weil descent attack of Gaudry, Hess and Smart, in *Proceedings of CT-RSA 2001*. LNCS, vol. 2020 (Springer, Berlin, 2001), pp. 308–318
[61] A. Menezes, S.A. Vanstone, The implementation of elliptic curve cryptosystems, in *Proceedings of AUSCRYPT 90*. LNCS, vol. 453 (Springer, Berlin, 1990), pp. 2–13
[62] V. Miller, Uses of elliptic curves in cryptography, in *Proceedings of CRYPTO 85*. LNCS, vol. 218 (Springer, Berlin, 1985), pp. 417–426
[63] P. Montgomery, Speeding the Pollard and elliptic curve methods of factorization. *Math. Comput.* **48**, 243–264 (1987)
[64] D. Naccache, N.P. Smart, J. Stern, Projective coordinates leak, in *Proceedings of EUROCRYPT 2004*. LNCS, vol. 3027 (Springer, Berlin, 2004), pp. 257–267
[65] National Institute of Standards and Technology, Recommended elliptic curves for federal government use. NIST special publication (1999). http://csrc.nist.gov/csrc/fedstandards.html
[66] National Institute of Standards and Technology, FIPS PUB 186-4: Digital Signature Standard (DSS). Federal Information Processing Standards (2013). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[67] National Security Agency, The case for elliptic curve cryptography, Oct 2005. https://web.archive.org/web/20051013062853/http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm?
[68] P.Q. Nguyen, I.E. Shparlinski, The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptogr.* **30**, 201–217 (2003)
[69] T. Oliveira, D.F. Aranha, J.L. Hernandez, F. Rodríguez-Henríquez, Fast point multiplication algorithms for binary elliptic curves with and without precomputation, in *Proceedings of SAC 2014*. LNCS, vol. 8781 (Springer, Berlin, 2014), pp. 324–344
[70] T. Oliveira, D.F. Aranha, J. López, F, Rodríguez-Henríquez, Improving the performance of the GLS254. Presentation at CHES 2016 rump session (2016)
[71] T. Oliveira, J. López, D.F. Aranha, F. Rodríguez-Henríquez, Two is the fastest prime: lambda coordinates for binary elliptic curves. *J. Cryptogr. Eng.* **4**(1), 3–17 (2014)
[72] T. Oliveira, J. López, F. Rodríguez-Henríquez, The Montgomery ladder on binary elliptic curves. Cryptology ePrint Archive, Report 2017/350 (2017). http://eprint.iacr.org/2017/350
[73] D. Page, Theoretical use of cache memory as a cryptanalytic side-channel. Cryptology ePrint Archive, Report 2002/169 (2002). http://eprint.iacr.org/
[74] G. Paoloni, How to benchmark code execution times on Intel IA-32 and IA-64 instruction set architectures. Technical report, Intel Corporation (2010)
[75] C. Petit, M. Kosters, A. Messeng, Algebraic approaches for the elliptic curve discrete logarithm problem over prime fields, in *Proceedings of PKC 2016*. LNCS, vol. 9615 (Springer, Berlin, 2016), pp. 3–18
[76] R. Sakai, K. Ohgishi, M. Kasahara, Cryptosystems based on pairing over elliptic curve (in Japanese), in *The 2001 Symposium on Cryptography and Information Security* (2001)
[77] R. Schroeppel, Cryptographic elliptic curve apparatus and method. US Patent 2002/6490352 B1 (2000)
[78] M. Scott, Optimal irreducible polynomials for $GF(2^m)$ arithmetic. Cryptology ePrint Archive, Report 2007/192 (2007). http://eprint.iacr.org/
[79] I. Semaev, Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031 (2004). http://eprint.iacr.org/2004/031
[80] I. Semaev, New algorithm for the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2015/310 (2015). http://eprint.iacr.org/2015/310
[81] J.A. Solinas, An improved algorithm for arithmetic on a family of elliptic curves, in *Proceedings of CRYPTO 97*. LNCS, vol. 1294 (Springer, Berlin, 1997), pp. 357–371
[82] J.A. Solinas, Efficient arithmetic on Koblitz curves. *Des. Codes Cryptogr.* **19**(2–3), 195–249 (2000)
[83] J. Tate, Endomorphisms of abelian varieties over finite fields. *Invent. Math.* **22**, 134–144 (1966)
[84] J. Taverne, A. Faz-Hernández, D.F. Aranha, F. Rodríguez-Henríquez, D. Hankerson, J. López, Software implementation of binary elliptic curves: impact of the carry-less multiplier on scalar multiplication, in *Proceedings of CHES 2011*. LNCS, vol. 6917 (Springer, Berlin, 2011), pp. 108–123
[85] W.R. Trost, G. Xu, On the optimal pre-computation of window τ-NAF for Koblitz curves. Cryptology ePrint Archive, Report 2014/664 (2014). http://eprint.iacr.org/
[86] Y. Tsunoo, E. Tsujihara, K. Minematsu, H. Miyauchi, Cryptanalysis of block ciphers implemented on computers with cache, in *International Symposium on Information Theory and Its Applications* (IEEE Information Theory Society, 2002), pp. 803–806
[87] M.D. Velichka, M.J. Jacobson Jr., A. Stein, Computing discrete logarithms in the Jacobian of high-genus hyperelliptic curves over even characteristic finite fields. *Math. Comput.* **83**(286), 935–963 (2014)
[88] A. Weimerskirch, C. Paar, Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, Report 2006/224 (2006). http://eprint.iacr.org/
[89] E. Wenger, P. Wolfger, Solving the discrete logarithm of a 113-Bit Koblitz curve with an FPGA cluster, in *Proceedings of SAC 2014*. LNCS, vol. 8781 (Springer, Berlin, 2014), pp. 363–379
[90] E. Wenger, P. Wolfger, Harder, better, faster, stronger: elliptic curve discrete logarithm computations on FPGAs. *J. Cryptogr. Eng.* **6**(4), 287–297 (2016)
[91] M.J. Wiener, R.J. Zuccherato, Faster attacks on elliptic curve cryptosystems, in *Proceedings of SAC 98*. LNCS, vol. 1556 (Springer, Berlin, 1999), pp. 190–200