Koblitz curves over quadratic fields

Thomaz Oliveira[1]⋆, Julio López[2]⋆⋆, Daniel Cervantes-Vázquez[1]⋆, and Francisco Rodríguez-Henríquez[1]⋆

[1] Computer Science Department, CINVESTAV-IPN
thomaz.figueiredo@gmail.com, francisco@cs.cinvestav.mx,
dcervantes@computacion.cs.cinvestav.mx
[2] Institute of Computing, University of Campinas
jlopez@ic.unicamp.br

Abstract. In this work, we retake an old idea that Koblitz presented in his landmark paper [50], 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 present 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.

1 Introduction

Let $(\mathbb{G}, \cdot)$ 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) in $\mathbb{G}$ is the computational problem of finding an integer x such that $g^x = h$. The integer x 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 when $\mathbb{G}$ is an elliptic curve E defined over a binary field, and where g is a point $P \in E(\mathbb{F}_q)$.

In 1985, Miller [59] and Koblitz [48] independently show 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 underlying hard computational problem. This was the birth of Elliptic Curve Cryptography (ECC),

⋆ The authors would like to thank CONACyT (project number 180421) for their funding of this research.

⋆⋆ The author was supported in part by the Intel Labs University Research Office and by a research productivity scholarship from CNPq Brazil.

which across the years has become one of the most intensively analyzed public key schemes in our discipline.[3]

In the abstract of his seminal paper, Koblitz [48] 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 [59], 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 one year before Miller and Koblitz independently presented their ECC proposal, Coppersmith had reported in [22],[4] 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 $0 < \alpha < 1$ and $c > 0$, denotes the expression, $exp\left((c + o(1))(\log q)^{\alpha}(\log \log q)^{1-\alpha}\right)$. Almost thirty years after, in February 2013, Joux [45] 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, 36]. 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 algorithm to solve it, is still the Pollard's Rho algorithm [29].

In the last decade however, inspired by the new lines of research presented by Semaev in 2004 [75], several researchers [30, 34, 46, 28, 76] (see also [29] for a comprehensive survey) have attempted to attack the DLP of all binary elliptic curves using summation-polynomial methods.[5] However, the current status

[3] See [51] for a historical recount of the first three decades of elliptic curve cryptography.

[4] Building on the work by [15].

[5] Sometimes also called Semaev's polynomials.

of these attempts are crucially based on ill-understood Gröbner basis assumptions, which in some cases have led to contradictory behaviors [40], even for tiny experiments [29].[6]

On the other side, it is now standard knowledge that the Pollard's 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 [9, 86]. Further, in the case of Koblitz curves, which are the main subject of this paper, one can apply the Frobenius endomorphism to speedup the Pollard rho algorithm by an extra $\sqrt{m}$ factor [33, 87]. 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*" [26].[7]

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 [49, 58]. However, when Menezes, Okamoto and Vanstone published their famous MOV attack [55, 56], 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 [18, 43, 44, 72]. 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, 12–14].

Nevertheless, disaster struck again about one decade later, when Joux [45] 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 [27, 32, 35, 39, 57] exploits the idea of finding an algebraic curve C of a relatively small genus g such that the Jacobian of C contains

[6] It is interesting to note that Semaev's original work in [75], described an attack that in principle can be applied not only to binary but to all elliptic curves [71].

[7] 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 $GF(2^n)$).

the target elliptic 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.

In general however, the GHS strategy is difficult to implement due to the large genus that a suitable curve C usually ends up having.[8] In fact, in [35, 57] 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 [54], 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 [83], it was estimated that when using the Enge-Gaudry algorithm [27], the cost of such computation is of around $1,736$ core-days.

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, 67, 66, 80] rank among the fastest Diffie-Hellman software benchmarked in the eBACS site [10] in various platforms. In particular, the software library announced in [66], 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.

Choosing side-channel resistant binary elliptic curves

Since the publication of Kocher's paper on differential power analysis [52], 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.

[8] 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 $E(\mathbb{F}_{2^m})$ [54].

In [11] (see also [19]), 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 [11] for a definition of each one of the desirable security properties that a safe curve should enjoy. Unfortunately, the authors in [11, 19] explicitly exclude binary elliptic curves from their analysis, without elaborating in the technical arguments to avoid them.

However, and as it will be discussed in the remaining of this paper, anomalous elliptic curves also known as Koblitz curves (see their definition in the next subsection), meet most if not all of the requirements specified for safe curves in [11]. Particularly, Koblitz curves arguably stand among the purest mathematical elliptic curve problems, which makes them as *fully rigid* as it gets. Similarly, Koblitz curves support the most efficient Montgomery ladder formulae that we know of (consisting of only 6 field multiplications per bit, which have the extra bonus of being amenable for parallelization). Moreover, Koblitz curves are amenable for right-to-left Montgomery ladders as discussed in [65]. This feature is especially valuable for the vast majority of protocols (such as the Diffie-Hellman protocol, ECDSA, 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, Koblitz curves do not meet the *completeness* criterion as defined in [11].

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 [50], 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 [49].

Koblitz curves defined over $\mathbb{F}_4$ were also proposed in [50]. Nevertheless, until now the research works dealing with standardized Koblitz curves in commercial use, such as the binary curves standardized by NIST [62, 64, 63] or the suite of elliptic curves supported by the TLS protocol [24, 16], 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, 17, 80, 85]).

We find 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 [53, 65, 67] 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 $a_0 + a_1 u$, 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, $d = a_0 b_0 + a_1 b_1 + ((a_0 + a_1) \cdot (b_0 + b_1) + a_0 b_0)u$. By carefully organizing the code associated to these arithmetic operations, one can greatly exploit the instruction-level parallelism of the pipelines that are available in contemporary high-end processors.

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 they were proposed by Koblitz in his 1991 seminal paper [50]. 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 ten years ago in [20, 25]. 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. For example, to the best of our knowledge, we report for the first time a 128-bit AVX implementation of the linear pass technique, which is useful against side-channel attacks.

The remaining of this paper is organized as follows. In §2 we formally introduce the family of Koblitz elliptic curves defined over $\mathbb{F}_4$. In §3 we analyze all the twelve 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 §4 and §5 a detailed description of the efficient implementation of the base and quadratic field arithmetic using vectorized instructions is given. We present in §6 the scalar multiplication algorithms used in this work, and we present in §7 the analysis and discussion of the results obtained by our software library. Finally, we draw our concluding remarks in §8.

2 Koblitz curves over $\mathbb{F}_4$

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

$$E_a : y^2 + xy = x^3 + a\gamma x^2 + \gamma, \qquad (1)$$

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 $E_1/\mathbb{F}_4$ are 4 and 6, respectively. For cryptographic purposes, one uses Eq. (1) operating over binary extension fields of the form $\mathbb{F}_q$, with $q = 4^m$, and m a prime number. The set of affine points $P = (x, y) \in \mathbb{F}_q \times \mathbb{F}_q$

that satisfy Eq. (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.

Table 1. Almost-prime group orders $\#E_a(\mathbb{F}_{4^m})$ with prime $m \in \{127, \ldots, 191\}$. Prime factors are underlined. The size (in bits) of the largest prime factor is presented in parenthesis

m	a	**Factorization of** $\#E_a(\mathbb{F}_{4^m})$
127	0	$0x4 \cdot \underline{0x1268F1298760419} \cdot$ $\underline{0xDE7D169BED4130151CD618CF5713077271FF51A4B1CFB75BF}$ (196)
127	1	$0x6 \cdot 0x41603EAF071 \cdot$ $\underline{0x29C4C778B6D2CD0FA36B3CA951A32DAC100C9C63576EEF7BF1F21}$ (209)
131	1	$0x6 \cdot 0x4267F1026F4F \cdot$ $\underline{0x2806BB97FB5F7C2F9E1EDE20BF59AC390DABBA7621D9A0F26AA1}$ (205)
137	0	$0x4 \cdot 0x763DB379950B73D200B971F1D \cdot$ $\underline{0x22A41FB03F2428B44188DD9FFEA796DC6D197A91BA21}$ (173)
137	1	$0x6 \cdot 0x4337925B3141B99447C1273 \cdot$ $\underline{0x289FE5979AC03A2E5CFCE8E6024FEF0863C633AE96A0DF}$ (182)
149	0	$0x4 \cdot 0x29B66B578C9FAEB \cdot$ $\underline{0x62322066993B57A8857E552587C80A567018483F2E493DBB7750AB7DB623}$ (239)
149	1	$0x6 \cdot \underline{0x1B73C442E8D} \cdot$ $\underline{0x637845F7F8BFAB325B85412FB54061F148B7F6E79AE11CC843ADE1470F7E4E29}$ (255)
157	0	$0x4 \cdot 0x499D09449B55C7D71FC18A2B0265785F \cdot$ $\underline{0x37A45BD5E114A84FCB8900BAEA9E731E0C4B3EDEC15F327}$ (186)
157	1	$0x6 \cdot 0xEECA8C4698A0916800B4E7 \cdot$ $\underline{0xB6F74A858FF10701D113E39259417F04CF038B297F3C6573F6E14F33}$ (224)
163	0	$0x4 \cdot \underline{0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF}\backslash$ $\underline{EA48D724AAB2045E5CFE286F8372017024DFF7BB3}$ (324)
167	1	$0x6 \cdot \underline{0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA}\backslash$ $\underline{D45C6A4A8565763007E9FEFA42E0EA9B9E8B7F3541}$ (331)
173	1	$0x6 \cdot 0xBA3DEF139 \cdot$ $\underline{0xEA9746EEF14E1638A503FA6FB739A623894A590811B6939A30D7A016E8A77815}\backslash$ $\underline{0084D9C4D6E0D}$ (308)
179	0	$0x4 \cdot 0x10C01861F3F8F0AC2767CD \cdot$ $\underline{0xF4882969C296A9493FEAA3C9F58DA166B76D3236BF15C2F10E2B0421F3F7E50DCC6F}$ (272)
181	0	$0x4 \cdot 0xCBB \cdot$ $\underline{0x141BF6E35420FDE10CF60620853943A20D5A91F2F5DDE75B04126F3100B191AF1}\backslash$ $\underline{E338F81FB8ED77C1C57BEF3}$ (348)
191	1	$0x6 \cdot 0x23D01 \cdot$ $\underline{0x4C3F9B376D369D04F03499007A43FE6460A012C86B2C575858EE9FC7F67A566813}\backslash$ $\underline{B39DA28DC9D58285BC07F8811}$ (362)

Since for each proper divisor r of s, $E(\mathbb{F}_{4^r})$ is a subgroup of $E(\mathbb{F}_{4^s})$, one has that $\#E(\mathbb{F}_{4^r})$ divides $\#E(\mathbb{F}_{4^s})$. Furthermore, by choosing prime extensions m, it is possible to find $E_a(\mathbb{F}_{4^m})$ 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 $m \in \{127, \dots, 191\}$.

The Frobenius map $\tau : E_a(\mathbb{F}_q) \to E_a(\mathbb{F}_q)$ 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 $P \in E_a(\mathbb{F}_q)$. By solving the equation $\tau^2 + 4 = \mu\tau$, the Frobenius map can be seen as the complex number $\tau = (\mu \pm \sqrt{-15})/2$.

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.[9] 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 [78], 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 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 [31].

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 [65] by slightly adapting 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 [77]. Given a width w, after running the regular τNAF algorithm we have $2^{2(w-1)-1}$ different digits.[10]

[9] Usually the order p is composite. Also, every prime factor of $p < r$ (see Table 1).

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

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

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

As a result, 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 addition savings obtained in the main iteration.

In addition, we must find efficient expressions of $\alpha_v = v \bmod \tau^w$. The method for searching the best expressions in Koblitz curves over $\mathbb{F}_2$ [81] 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. 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. In Table 2 we present the α_v representatives along with the operations required to generate the multiples of the base point.[11]

Therefore, 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.

2.3 Security analysis against the GHS attack

Since the Koblitz curves defined over $E_a(\mathbb{F}_{4^m})$ operate over quadratic extensions fields, it is conceivable that Weil descent attacks [35, 39] could possibly be efficiently applied on these curves. However, Menezes and Qu showed in [57] 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 $[160, \ldots, 600]$. Further, a specialized analysis for binary curves defined over fields of the form $\mathbb{F}_{4^m}$ reported in [37], proved that the only vulnerable prime extension in the range $[80, \ldots, 256]$ 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 Section 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 next.

[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.

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 $R(x_0) = \frac{p(x_0)}{q(x_0)}$ and $\gcd(p(x_0), q(x_0)) = 1$, then the degree of ϕ is $\max\{\deg p(x_0), \deg(q(x_0))\}$. 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 there exists an isomorphism between them.

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 [79, Theorem 1]. This fact helps to classify elliptic curves by isogeny classes, i.e. by their point cardinality.

3.2 The twelve ordinary elliptic curves over $\mathbb{F}_4$

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

Table 3. The 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 $\gamma \in \mathbb{F}_4 \setminus \mathbb{F}_2$.

	$E_{(a,b)}/\mathbb{F}_4 : y^2 + xy = x^3 + ax^2 + b$			
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).	$(\gamma,1)$, $(\gamma^2,1)$.

Notice that the Koblitz curves described in Section 2 corresponds to the 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 to the isogeny classes 0 and 3.

The curves $E_{(1,1)}$ and $E_{(0,1)}$ are the only two members of the 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 Eq. (1) with $a = 0$. This observation 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 $\#E_{(0,1)}(\mathbb{F}_{2^m}) \approx \#E_{(1,1)}(\mathbb{F}_{2^m}) \approx 2^m$. 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. We conclude that the isogeny class 0 is of little or no cryptographic value.

3.3 A Novel endomorphism for the isogeny Class 3

The curves $E_{(\gamma,1)}$ and $E_{(\gamma^2,1)}$ are the only two members of the 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 $\tau_2 : E_{(a,b)}(\mathbb{F}_q) \to E_{(a',b')}(\mathbb{F}_q)$ 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, $\phi_0 : E_{(\gamma,1)} \to E_{(\gamma^2,1)}$ and $\phi_1 : E_{(\gamma^2,1)} \to E_{(\gamma,1)}$ 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 Figure 1, for each one of the curves in the 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 [67], the endomorphisms $\bar{\tau}_0$ and $\bar{\tau}_1$ can be written as, $\bar{\tau}_0(x, \lambda) = (x^2, \lambda^2 + \gamma)$ and $\bar{\tau}_0(x, \lambda) = (x^2, \lambda^2 + \gamma^2)$, respectively. Using projective λ-coordinates they become $\bar{\tau}_0(x, \lambda, z) = (x^2, \lambda^2 + \gamma \cdot z^2, z^2)$ and $\bar{\tau}_0(x, \lambda) = (x^2, \lambda^2 + \gamma^2 \cdot z^2, z^2)$, respectively.

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

$$\begin{tikzcd}
E_{(\gamma,1)} \arrow[d, "\tau_2"'] \\
E_{(\gamma^2,1)} \arrow[u, bend right=60, "\phi_0"']
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
E_{(\gamma^2,1)} \arrow[d, "\tau_2"'] \\
E_{(\gamma,1)} \arrow[u, bend right=60, "\phi_1"']
\end{tikzcd}$$

Since $\bar{\tau}_0$ and $\bar{\tau}_1$ satisfy the same properties we will use in the following $\bar{\tau}$ to refer 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 to τ. As it will be further discussed in §7, this computational saving induces an 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 the 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, 17, 80, 81] for performing the τw-NAF scalar recoding.

We computed the cardinality of the elliptic curves belonging to the 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 the class 3 has the following integer factorization:

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

The largest prime factor above has a size of 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 10 bits less of security than the number $\lfloor \frac{m}{2} \rfloor$).

4 Base field arithmetic

From this part onwards, we introduce practical aspects of the Koblitz curves presented in the previous sections. More specifically, we describe the techniques for an efficient software implementation of 128-bit secure scalar multiplication algorithms over selected Koblitz curves defined over $\mathbb{F}_4$. In this work, we focused on high-end 64-bit processors embedded with a 64-bit carry-less multiplication instruction and 128-bit registers that store and simultaneously process two 64-bit words [1, 41].

We based our curve selection on two basic 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 §1 for further discussion). Moreover, it is equipped with a faster endomorphism when compared with class-1 and -2 curves. For other scenarios, we suggest a 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 $f(x)$ depends on the architecture to be implemented as it was extensively discussed in [74].

For our field extension choices, we do not have irreducible trinomials in $\mathbb{F}_2[x]$. The immediate solution is to construct the field through irreducible pentanomials. The problem here is that pentanomials make the modular reduction too costly if we compare it 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 they are shifts not divisible by the word size (64 bits in our target architecture).

As a consequence, we resorted to the redundant trinomials strategy introduced in [20, 25]. 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 software platform counts with a native 64-bit carry-less multiplier, an efficient representation of the field elements must have at most 192 bits, *i.e.* 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 §5 for more information).

(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 it avoids additions between words, which requires a sequence of left and right shifts to align the bits.

(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 [38]. Therefore, this property saves one shift and one addition in each reduction step, as the shift by 1 is avoided.

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 field operations, however there was no better options within 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 [17].

5 Quadratic field arithmetic

In this section, the basic arithmetic operations in the quadratic field are presented. As usual, the quadratic field was constructed by the degree two monic polynomial $h(u) = u^2 + u + 1$ and its elements are represented as $a_0 + a_1 u$, with $a_0, a_1 \in \mathbb{F}_{2^m}$.

5.1 Register allocation

The first aspect to be considered for implementing an efficient quadratic field arithmetic is how to allocate the binary representation of the field elements into the architecture's available registers. In our case, we have to store two polynomials of 192 bits into 128-bit registers in such way that it allows a fast modular reduction and, at the same time, generates a minimum overhead in the two main arithmetic operations, namely the field multiplication and squaring.

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 in three 64-bit words (A-C, D-F). Also, let us 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. §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, there still exists some overhead in the multiplication and squaring arithmetic operations, 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, as discussed next.

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 by three 64-bit words. The analysis of the Karatsuba algorithm cost for different word sizes was presented in [84]. 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 twelve additions.

The algorithm requires six carry-less instructions, six vectorized `xors` 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 (twelve vectorized `xors` and six byte interleaving instructions) and at the quadratic field level (six vectorized `xors`). Also, we must consider the reorganization of the registers in order to proceed with the modular reduction (six vectorized `xors`).

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. That is, the real and the imaginary part 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. §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 `xors`, and three such steps are required.

Nonetheless, for the $\mathbb{F}_{4^{163}}$ case, we have to compute four shifts and four `xors` per step. In the total cost, the modular reduction in this field requires three extra shifts and `xors` when compared with the same operation over $\mathbb{F}_{4^{149}}$. Although this difference does not appear to be significant, its impact will be seen in the total cost of the scalar multiplication algorithm (see §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 [38, §2.3.4]. The pre-processing phase is straightforward, 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 `xors` and three bitwise shifts.

5.5 Inversion

The inversion operation is computed via the Itoh-Tsujii method [42]. 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 [21]. 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 section we briefly discuss single-core algorithms that compute a timing-resistant scalar multiplication function over Koblitz curves. The faster way is to compute the function via width-w τ-and-add algorithms [38] with their right-to-left and left-to-right variants. A more conservative approach is to perform the point multiplication with Montgomery ladders [60]. This method disregards the efficient τ endomorphism but takes profit of the Koblitz curves 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] In addition, we need to compute the width w-τNAF representation of the scalar k and perform linear passes (cf.§6.3) in the accumulators in order to avoid cache-based timing attacks [82, 69]. The method is shown in Algorithm 1.

The main advantage of this method is that the sensitive data is indirectly reflected in the points P_{ρ_i}, which are only read and then added to the unique

[14] On recent Intel architectures, this instruction is denominated `pshufd` [41].

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

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

```
Input: A Koblitz curve $E/\mathbb{F}_{4^m}$, point $P \in E(\mathbb{F}_{4^m})$ of order $r$, $k \in \mathbb{Z}[\tau] = k_0 + k_1\tau$
Output: $Q = kP$
1: Compute the width-$w$ regular $\tau$-NAF of $k_0 + k_1\tau$ as $\sum_{i=0}^{l} \rho_i \tau^{i(w-1)}$
2: Pre-compute the $2^{w-1}$ $(2^{2(w-1)-1})$ points $P_{\rho_i} \leftarrow \rho_i P$ 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 $P_{\rho_i}$
7:     $Q \leftarrow Q \pm P_{\rho_i}$
8: end for
9: return $Q = kP$
```

accumulator Q. As a consequence, just one linear pass per iteration is required, one step before reading P_{ρ_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 look-up tables could leak information about the scalar k bits.

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 it (ideally) does not interact with 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 [78]. If desired, the scalar can be retrieved as an integer in $\mathbb{Z}_r$ as discussed in §2.1.

6.2 Right-to-left τ-and-add

The other variant processes the scalar k from the least to the most significant digit. Taking advantage of the τ endomorphism, the GLV method is brought to its full extent. This approach is presented in Algorithm 2.

Here, we have to perform a post-computation in the accumulators instead of precomputing the points P_{ρ_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 look-ups instead of performing squarings multiple times. The downside of this algorithm is that the accumulators carry sensitive information about on the scalar k digits. Moreover, they are read and written. As a result, it is necessary to apply the linear pass algorithm over the accumulators Q_{ρ_i} twice per iteration.

6.3 Linear pass

The linear pass is a method designed to protect sensitive information against side-channel attacks associated with the CPU cache access patterns. Let us consider an array A of size l. Before reading a value $A[i]$, with $i \in \{0, \dots, l-1\}$,

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

```
Input: A Koblitz curve $E/\mathbb{F}_{4^m}$, point $P \in E(\mathbb{F}_{4^m})$ of order $r$, $k \in \mathbb{Z}[\tau] = k_0 + k_1\tau$
Output: $Q = kP$
 1: Compute the width-$w$ regular $\tau$-NAF of $k_0 + k_1\tau$ as $\sum_{i=0}^{l} \rho_i \tau^{i(w-1)}$
 2: Initialize the $2^{w-1}$ ($2^{2(w-1)-1}$) accumulators $Q_{\rho_i} \leftarrow \mathcal{O}$ for class-3 (class-1,2) $E$
 3: for $i = 0$ to $l$ do
 4:     Perform a linear pass to recover $Q_{\rho_i}$
 5:     $Q_{\rho_i} \leftarrow Q_{\rho_i} \pm P$
 6:     Perform a linear pass to store $Q_{\rho_i}$
 7:     $P \leftarrow \tau^{w-1}(P)$
 8: end for
 9: $Q \leftarrow \mathcal{O}$
10: Post-compute the $2^{w-1}$ ($2^{2(w-1)-1}$) points $Q \leftarrow \sum \rho_i Q_{\rho_i}$ for $E$ in class-3 (class-1,2)
11: return $Q = kP$
```

the linear pass technique reads the entire array A but only stores, usually into an output register, the requested data $A[i]$.

In that way, 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 is stored in 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 that depends on 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.

Conditional swap The conditional swap is a technique usually employed in Montgomery ladders to hide the access order of the two accumulators R_0, R_1, which is directly related to the bits of the scalar k. This technique can be easily 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, there still a possibility of extending the conditional swap to 2^l points in l steps, with $l > 1$. However, the number of `xors` required would result in a procedure more expensive than the double linear pass discussed above. There is still a possibility of performing the conditional swap only in the lowest 2^{i-1} array A values in each step i, for $1 \leq i \leq l$. However, this would require bookkeeping the state of the array.

6.4 Montgomery ladder

We considered the implementation of the left-to-right and right-to-left Montgomery ladder over the class-3 Koblitz curve over $\mathbb{F}_{4^{163}}$, which was proposed as our conservative option. The motivation for implementing this algorithm was to exploit the convenient curve b-parameter, which is equal to 1. In this case, the left-to-right doubling formula only requires three field squarings and one multiplication. Moreover, we selected our input point P x-coordinate $= x^{64}u$. The field multiplication by this special element requires only sixteen logical vector instructions. For the right-to-left variant, the b-parameter allows us to store only one set of points since the pre-computed values μ_0 and μ_1 is the same (see [68] for more details on the Montgomery formulas for binary curves).

In this algorithm, the scalar k was constructed as follows. First, 40 bytes K_i was chosen at random from $[0, 255]$. Then, we processed the most and least significant byte as

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

Next, the scalar was 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 was always equal to 1, avoiding any inconsistencies during the Montgomery addition left-to-right and right-to-left formulas.

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 [8].

Error margin The analysis in [70] 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. Nonetheless, for simpler functions such as most of the finite field operations, it is recommended to consider this margin.

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 §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

$$r = \texttt{0xC8B90A95C20EE5BBC91D671B0CEFED2EA701F5CE}\backslash$$
$$\texttt{9AAA522F37A4E0D020A19EBBDC1D0437C458139}$$

of approximately 315 bits. In addition, throughout our τ-and-add scalar multiplication implementation, we represented the points in λ-affine [47, 73] and λ-projective [67] 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 3 clock cycles. The column 'ratio op/m' shows the ratio between the timings of the presented operation and the field multiplication.

Table 4. Timings (in clock cycles) 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	6,600	126.92	6,300	92.65
reduction modulo $f_m(x)$	452	8.69	432	6.35

The ratio squaring/multiplication is relatively high when compared with other binary curves such as GLS-254 [67]. This is because here the trinomials $g_m(x)$ do not admit a reduction specially crafted for the squaring operation. 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 [61], we did not work with look-up tables for computing the different Itoh-Tsujii multisquaring operations, but applied consecutive field squarings. Finally, the result of having a redundant trinomial which do not comply with criteria R2 (see §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.

Table 5. Timings (in clock cycles) 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 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 [16] is 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 outperforms 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 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 width-w represent a different number of pre/post-computed points when comparing the two curve classes, since the $\tau, \bar{\tau}$ endomorphisms are different and therefore the recoding algorithm.

For the sake of comparison, Table 6 includes the ratio clock cycles per bit. This is because it wouldn't be fair to simply compare the absolute timings of each curve, since the security provided by the class-3 curve is higher than our proposed class-1 curve. The ratio considers only the number of random bits processed by the algorithm. 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.

[16] 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 (in clock cycles) for the τ-and-add scalar multiplication on $E_{(0,u)}/\mathbb{F}_{4^{149}}$ and $E_{(u,1)}/\mathbb{F}_{4^{163}}$

Left-to-right τ-and-add	$E_{(0,u)}/\mathbb{F}_{4^{149}}$		$E_{(u,1)}/\mathbb{F}_{4^{163}}$	
	cost (cc)	ratio cycles/bit	cost (cc)	ratio cycles/bit
$w = 2$	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	$E_{(0,u)}/\mathbb{F}_{4^{149}}$		$E_{(u,1)}/\mathbb{F}_{4^{163}}$	
	cost (cc)	ratio cycles/bit	cost (cc)	ratio cycles/bit
$w = 2$	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

The above results show that for the class-1 case, the cost of the pre- and post-computation increases rapidly with the windows w widths. The increase of point pre-computation timings between widths $w = 2$ and 3 is of about 86.93%, while the difference between widths $w = 3$ and 4 is about 168.97%. This is because the number of points to be computed quadruple 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.

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

The same issue does not occur in the class-3 curve, since the 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 better 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 the following table, the results for the Montgomery ladder algorithms.

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

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

The savings in the left-to-right Montgomery addition formulas resulted in compatible timings when compared with the τ-and-add algorithms with $w \leq 3$. The advantage of the 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 algorithm results are not comparable with the costs presented in Table 6, since it is employed in the fixed-point scenario.

7.4 Comparison

Here, 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.

Table 9. A comparative between state-of-the-art software implementations of 128-bit secure timing-resistant scalar multiplication. The timings were measured in the Haswell platform and are given in clock cycles

Curve/Method	Timing
Koblitz over $\mathbb{F}_{2^{283}}$ (τ-and-add, 5-τ-NAF) [65]	99,000
GLS over $\mathbb{F}_{4^{127}}$ (2-GLV double-and-add, 5-NAF) [66]	48,300
Twisted Edwards over $\mathbb{F}_{(2^{127}-1)^2}$ (double-and-add) [23]	56,000
Kummer genus-2 over $\mathbb{F}_{2^{127}-1}$ (Kummer ladder) [7]	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 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 [65] by 20.29%. On binary fields, the GLS implementation in [66] outperforms our faster implementation by 38.80%. The works in [23] and [7] are about 29.04% and 23.27% faster, respectively. Finally, our $E_{(u,1)}/\mathbb{F}_{4^{163}}$ curve is only 6,000 cycles more expansive than the K-283 implementation in [65], but offers 15 extra bits of security. Given that both 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 completely taxonomy of ordinary elliptic curves defined over the extension fields $\mathbb{F}_{4^m}$, with good cryptographic properties. We introduced a family of Koblitz-like elliptic curves, whose rich structure admits novel and more efficient endomorphisms than the traditional Frobenius endomorphism associated with classical Koblitz curves.

We presented novel 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 distinguish 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.

As a result of those computational optimizations, we achieved the fastest software timings for Koblitz curves at the 128-bit security level. Furthermore, our reported benchmarks show that our scalar multiplication algorithms are performance competitive with state-of-the-art scalar multiplication implementations on binary and prime elliptic curves.

Acknowledgments

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, and F. Rodríguez-Henríquez. Faster Implementation of Scalar Multiplication on Koblitz Curves. In *Proceedings of LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 177–193. Springer, 2012.
4. D. F. Aranha, J. López, and D. Hankerson. Efficient Software Implementation of Binary Field Arithmetic Using Vector Instruction Sets. In *Proceedings of LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 144–161. Springer, 2010.
5. A. U. Ay, E. Öztürk, F. Rodríguez-Henríquez, and E. Savaş. Design and Implementation of a Constant-time FPGA Accelerator for Fast Elliptic Curve Cryptography. In *ReConFig 2016*, pages 1–8. IEEE, 2016.

6. R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. In *Proceedings of EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 1–16. Springer, 2014.
7. D. J. Bernstein, C. Chuengsatiansup, T. Lange, and P. Schwabe. Kummer strikes back: New DH speed records. In *Proceedings of ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 317–337. Springer, 2014.
8. D. J. Bernstein and T. L. (editors). eBACS: ECRYPT Benchmarking of Cryptographic Systems. `http://bench.cr.yp.to`, accessed 14 Dec 2016.
9. D. J. Bernstein, S. Engels, T. Lange, R. Niederhagen, C. Paar, P. Schwabe, and R. Zimmermann. Faster discrete logarithms on FPGAs. Cryptology ePrint Archive, Report 2016/382, 2016. `http://eprint.iacr.org/2016/382`.
10. D. J. Bernstein and T. Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems. accessed 12 Dec 2016. `http://bench.cr.yp.to`.
11. D. J. Bernstein and T. Lange. SafeCurves: choosing safe curves for elliptic-curve cryptography. `http://safecurves.cr.yp.to`, accessed 14 Dec 2016.
12. J. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, and 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*, volume 5209 of *LNCS*, pages 297–315. Springer, 2008.
13. J. Beuchat, J. Detrey, N. Estibals, E. Okamoto, and F. Rodríguez-Henríquez. Fast Architectures for the η_T Pairing over Small-Characteristic Supersingular Elliptic Curves. *IEEE Trans. Computers*, 60(2):266–281, 2011.
14. J. Beuchat, E. López-Trejo, L. Martínez-Ramos, S. Mitsunari, and F. Rodríguez-Henríquez. Multi-core Implementation of the Tate Pairing over Supersingular Elliptic Curves. In *Proceedings of CANS 2009*, volume 5888 of *LNCS*, pages 413–432. Springer, 2009.
15. I. F. Blake, R. Fuji-Hara, R. C. Mullin, and S. A. Vanstone. Computing logarithms in finite fields of characteristic two. *SIAM Journal on Algebraic and Discrete Methods*, 5:276–285, 1984.
16. S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and 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`.
17. M. Bluhm and S. Gueron. Fast software implementation of binary elliptic curve cryptography. *J. Cryptographic Engineering*, 5(3):215–226, 2015.
18. D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.
19. J. W. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: an efficiency and security analysis. *Journal of Cryptographic Engineering*, 6(4):259–286, 2016.
20. R. P. Brent and 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*, page 212, 2003.
21. N. M. Clift. Calculating optimal addition chains. *Computing*, 91(3):265–284, 2011.
22. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Information Theory*, 30(4):587–593, 1984.
23. C. Costello and P. Longa. Four($\mathbb{Q}$): Four-Dimensional Decompositions on a ($\mathbb{Q}$)-curve over the Mersenne Prime. In *Proceedings of ASIACRYPT 2015*, volume 9452 of *LNCS*, pages 214–235. Springer, 2015.

24. T. Dierks and 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`.
25. C. Doche. Redundant Trinomials for Finite Fields of Characteristic 2. In *Proceedings of ACISP 2005*, volume 3574 of *LNCS*, pages 122–133. Springer, 2005.
26. ECRYPT II. Ecrypt II yearly report on algorithms and keysizes (2011-2012). Katholieke Universiteit Leuven (KUL), 2012. `http://www.ecrypt.eu.org/`.
27. A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, 102:83–103, 2002.
28. J. Faugère, L. Perret, C. Petit, and G. Renault. Improving the Complexity of Index Calculus Algorithms in Elliptic Curves over Binary Fields. In *Proceedings of EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 27–44. Springer, 2012.
29. S. D. Galbraith and P. Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Des. Codes Cryptography*, 78(1):51–72, 2016.
30. S. D. Galbraith and S. W. Gebregiyorgis. Summation polynomial algorithms for elliptic curves in characteristic two. In *Proceedings of INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 409–427. Springer, 2014.
31. S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves. In *Proceedings of EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 518–535. Springer, 2009.
32. S. D. Galbraith and N. P. Smart. A Cryptographic Application of Weil Descent. In *Proceedings of Cryptography and Coding*, volume 1746 of *LNCS*, pages 191–200. Springer, 1999.
33. R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Improving the parallelized pollard lambda search on anomalous binary curves. *Math. Comput.*, 69(232):1699–1705, 2000.
34. 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.
35. P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15:19–46, 2002.
36. R. Granger, T. Kleinjung, and J. Zumbrägel. On the Powers of 2. Cryptology ePrint Archive, Report 2014/300, 2014. `http://eprint.iacr.org/2014/300`.
37. D. Hankerson, K. Karabina, and A. Menezes. Analyzing the Galbraith-Lin-Scott Point Multiplication Method for Elliptic Curves over Binary Fields. *Computers, IEEE Transactions on*, 58(10):1411 – 1420, 2009.
38. D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag, Secaucus, NJ, USA, 2003.
39. F. Hess. Generalising the GHS Attack on the Elliptic Curve Discrete Logarithm Problem. *LMS Journal of Computation and Mathematics*, 7:167–192, 2004.
40. Y.-J. Huang, C. Petit, N. Shinohara, and T. Takagi. On Generalized First Fall Degree Assumptions. Cryptology ePrint Archive, Report 2015/358, 2015. `http://eprint.iacr.org/2015/358`.
41. Intel Corporation. Intel 64 and IA-32 Architectures Software Developers Manual 253665-064US., 2017.
42. T. Itoh and S. Tsujii. A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) Using Normal Bases. *Inf. Comput.*, 78(3):171–177, 1988.
43. A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Proceedings of ANTS-IV*, volume 1838 of *LNCS*, pages 385–394. Springer, 2000.
44. A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, 2004.

45. A. Joux. A New Index Calculus Algorithm with Complexity $L(1/4+o(1))$ in Small Characteristic. In *Proceedings of SAC 2013*, volume 8282 of *LNCS*, pages 355–379. Springer, 2014.
46. K. Karabina. Point Decomposition Problem in Binary Elliptic Curves. Cryptology ePrint Archive, Report 2015/319, 2015. `http://eprint.iacr.org/2015/319`.
47. E. Knudsen. Elliptic Scalar Multiplication Using Point Halving. In *Proceedings of ASIACRYPT 99*, volume 1716 of *LNCS*, pages 135–149. Springer Berlin Heidelberg, 1999.
48. N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of computation*, 48:203–9, 1987.
49. N. Koblitz. Constructing Elliptic Curve Cryptosystems in Characteristic 2. In *Proceedings of CRYPTO 90*, volume 537 of *LNCS*, pages 156–167, 1990.
50. N. Koblitz. CM-Curves with Good Cryptographic Properties. In *Proceedings of CRYPTO 1991*, volume 576 of *LNCS*, pages 279–287. Springer, 1991.
51. N. Koblitz and A. Menezes. A Riddle Wrapped in an Enigma. Cryptology ePrint Archive, Report 2015/1018, 2015. `http://eprint.iacr.org/2015/1018`.
52. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proceedings of CRYPTO 99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
53. P. Longa and F. Sica. Four-Dimensional Gallant-Lambert-Vanstone Scalar Multiplication. *J. Cryptology*, 27(2):248–283, 2014.
54. M. Maurer, A. Menezes, and 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*, volume 2247 of *LNCS*, pages 195–213. Springer, 2001.
55. A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39:1639–1646, 1993.
56. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *STOC 91*, pages 80–89. ACM, 1992.
57. A. Menezes and M. Qu. Analysis of the Weil Descent Attack of Gaudry, Hess and Smart. In *Proceedings of CT-RSA 2001*, volume 2020 of *LNCS*, pages 308–318. Springer, 2001.
58. A. Menezes and S. A. Vanstone. The Implementation of Elliptic Curve Cryptosystems. In *Proceedings of AUSCRYPT 90*, volume 453 of *LNCS*, pages 2–13. Springer, 1990.
59. V. Miller. Uses of Elliptic Curves in Cryptography. In *Proceedings of CRYPTO 85*, volume 218 of *LNCS*, pages 417–426. Springer, 1985.
60. P. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48:243–264, 1987.
61. D. Naccache, N. P. Smart, and J. Stern. Projective Coordinates Leak. In *Proceedings of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 257–267. Springer Berlin Heidelberg, 2004.
62. National Institute of Standards and Technology. Recommended Elliptic Curves for Federal Government Use. NIST Special Publication, 1999. `http://csrc.nist.gov/csrc/fedstandards.html`.
63. National Institute of Standards and Technology. FIPS PUB 186-4: Digital Signature Standard (DSS). Federal Information Processing Standards, 2013. `nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`.
64. National Security Agency. The case for elliptic curve cryptography, October 2005. `tinyurl.com/NSAandECC`.

65. T. Oliveira, D. F. Aranha, J. L. Hernandez, and F. Rodríguez-Henríquez. Fast Point Multiplication Algorithms for Binary Elliptic Curves with and without Precomputation. In *Proceedings of SAC 2014*, volume 8781 of *LNCS*, pages 324–344. Springer, 2014.
66. T. Oliveira, D. F. Aranha, J. López, and F. Rodríguez-Henríquez. Improving the performance of the GLS254. Presentation at CHES 2016 rump session, 2016.
67. T. Oliveira, J. López, D. F. Aranha, and F. Rodríguez-Henríquez. Two is the fastest prime: lambda coordinates for binary elliptic curves. *J. Cryptographic Engineering*, 4(1):3–17, 2014.
68. T. Oliveira, J. López, and F. Rodríguez-Henríquez. The montgomery ladder on binary elliptic curves. Cryptology ePrint Archive, Report 2017/350, 2017. `http://eprint.iacr.org/2017/350`.
69. D. Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. Cryptology ePrint Archive, Report 2002/169, 2002. `http://eprint.iacr.org/`.
70. G. Paoloni. How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures. Technical report, Intel Corporation, 2010.
71. C. Petit, M. Kosters, and A. Messeng. Algebraic Approaches for the Elliptic Curve Discrete Logarithm Problem over Prime Fields. In *Proceedings of PKC 2016*, volume 9615 of *LNCS*, pages 3–18. Springer, 2016.
72. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairing over Elliptic Curve (in Japanese). In *The 2001 Symposium on Cryptography and Information Security*, 2001.
73. R. Schroeppel. Cryptographic elliptic curve apparatus and method, 2000. US patent 2002/6490352 B1.
74. M. Scott. Optimal Irreducible Polynomials for $GF(2^m)$ Arithmetic. Cryptology ePrint Archive, Report 2007/192, 2007. `http://eprint.iacr.org/`.
75. 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`.
76. 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`.
77. J. A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In *Proceedings of CRYPTO 97*, volume 1294 of *LNCS*, pages 357–371. Springer Berlin Heidelberg, 1997.
78. J. A. Solinas. Efficient Arithmetic on Koblitz Curves. *Designs, Codes and Cryptography*, 19(2-3):195–249, 2000.
79. J. Tate. Endomorphisms of abelian varieties over finite fields. *Inventiones Mathematicae*, 22:134–144, 1966.
80. J. Taverne, A. Faz-Hernández, D. F. Aranha, F. Rodríguez-Henríquez, D. Hankerson, and J. López. Software Implementation of Binary Elliptic Curves: Impact of the Carry-less Multiplier on Scalar Multiplication. In *Proceedings of CHES 2011*, volume 6917 of *LNCS*, pages 108–123. Springer, 2011.
81. W. R. Trost and 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/`.
82. Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi. Cryptanalysis of Block Ciphers Implemented on Computers with Cache. In *International Symposium on Information Theory and Its Applications*, pages 803–806. IEEE Information Theory Society, 2002.

83. M. D. Velichka, M. J. J. Jr., and A. Stein. Computing discrete logarithms in the Jacobian of high-genus hyperelliptic curves over even characteristic finite fields. *Math. Comput.*, 83(286), 2014.
84. A. Weimerskirch and C. Paar. Generalizations of the Karatsuba Algorithm for Efficient Implementations. Cryptology ePrint Archive, Report 2006/224, 2006. `http://eprint.iacr.org/`.
85. E. Wenger and P. Wolfger. Solving the Discrete Logarithm of a 113-Bit Koblitz Curve with an FPGA Cluster. In *Proceedings of SAC 2014*, volume 8781 of *LNCS*, pages 363–379. Springer, 2014.
86. E. Wenger and P. Wolfger. Harder, better, faster, stronger: elliptic curve discrete logarithm computations on FPGAs. *J. Cryptographic Engineering*, 6(4):287–297, 2016.
87. M. J. Wiener and R. J. Zuccherato. Faster Attacks on Elliptic Curve Cryptosystems. In *Proceedings of SAC 98*, volume 1556 of *LNCS*, pages 190–200. Springer, 1999.