The Realm of the Pairings

Diego F. Aranha[1], Paulo S. L. M. Barreto[2⋆],
Patrick Longa[3], and Jefferson E. Ricardini[2]

[1] Department of Computer Science, University of Brasília, Brazil.
dfaranha@unb.br
[2] Departamento de Engenharia de Computação e Sistemas Digitais,
Escola Politécnica, University of São Paulo, Brazil.
{pbarreto,jricardini}@larc.usp.br
[3] Microsoft Research,
One Microsoft Way, Redmond, USA.
plonga@microsoft.com

Abstract. Bilinear maps, or pairings, initially proposed in a cryptologic context for cryptanalytic purposes, proved afterward to be an amazingly flexible and useful tool for the construction of cryptosystems with unique features. Yet, they are notoriously hard to implement efficiently, so that their effective deployment requires a careful choice of parameters and algorithms. In this paper we review the evolution of pairing-based cryptosystems, the development of efficient algorithms and the state of the art in pairing computation, and the challenges yet to be addressed on the subject, while also presenting some new algorithmic and implementation refinements in affine and projective coordinates.

Keywords: pairing-based cryptosystems, efficient algorithms.

1 Introduction

Bilinear maps, or *pairings*, between the (divisors on the) groups of points of certain algebraic curves over a finite field, particularly the Weil pairing [94] and the Tate (or Tate-Lichtenbaum) pairing [45], have been introduced in a cryptological scope for destructive cryptanalytic purposes, namely, mapping the discrete logarithm problem on those groups to the discrete logarithm problem on the multiplicative group of a certain extension of the base field [66, 46]: while the best generic classical (non-quantum) algorithm for the discrete logarithm problem on the former groups may be exponential, in the latter case subexponential algorithms are known, so that such a mapping may yield a problem that is asymptotically easier to solve.

It turned out, perhaps surprisingly, that these same tools have a much more relevant role in a constructive cryptographic context, as the basis for the definition of cryptosystems with unique properties. This has been shown in the seminal works on identity-based non-interactive authenticated key agreement by

⋆ Supported by CNPq research productivity grant 306935/2012-0.

Sakai, Ohgishi and Kasahara [84], and on one-round tripartite key agreement by Joux [56], which then led to an explosion of protocols exploring the possibilities of *identity-based cryptography* and many other schemes, with ever more complex features.

All this flexibility comes at a price: pairings are notoriously expensive in implementation complexity and processing time (and/or storage occupation, in a trade-off between time and space requirements). This imposes a very careful choice of algorithms and curves to make them really practical. The pioneering approach by Miller [67, 68] showed that pairings could be computed in polynomial time, but there is a large gap from there to a truly efficient implementation approach.

Indeed, progress in this line of research has not only revealed theoretical bounds on how efficiently a pairing can be computed in the sense of its overall order of complexity [93], but actually the literature has now very detailed approaches on how to attain truly practical, extremely optimized implementations that cover all operations typically found in a pairing-based cryptosystem, rather than just the pairing itself [4, 80]. One can therefore reasonably ask how far this trend can be pushed, and how "notoriously expensive" pairings really are (or even whether they really are as expensive as the folklore pictures them).

Our contribution: In this paper we review the evolution of pairing-based cryptosystems, the development of efficient algorithms for the computation of pairings and the state of the art in the area, and the challenges yet to be addressed on the subject.

Furthermore, we provide some new refinements to the pairing computation in affine and projective coordinates over ordinary curves, perform an up-to-date analysis of the best algorithms for the realization of pairings with special focus on the 128-bit security level and present a very efficient implementation for x64 platforms.

Organization: The remainder of this paper is organized as follows. Section 2 introduces essential notions on elliptic curves and bilinear maps for cryptographic applications, including some of the main pairing-based cryptographic protocols and their underlying security assumptions. Section 3 reviews the main proposals for pairing-friendly curves and the fundamental algorithms for their construction and manipulation. In Section 4, we describe some optimizations to formulas in affine and projective coordinates, carry out a performance analysis of the best available algorithms and discuss benchmarking results of our high-speed implementation targeting the 128-bit security level on various x64 platforms. We conclude in Section 5.

2 Preliminary concepts

Let $q = p^m$. An *elliptic curve* $E/\mathbb{F}_q$ is a smooth projective algebraic curve of genus one with at least one point. The affine part satisfies an equation of the form

$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ where $a_i \in \mathbb{F}_q$. Points on E are affine points $(x, y) \in \mathbb{F}_q^2$ satisfying the curve equation, together with an additional point at infinity, denoted ∞. The set of curve points whose coordinates lie in a particular extension field $\mathbb{F}_{q^k}$ is denoted $E(\mathbb{F}_{q^k})$ for $k > 0$ (note that the a_i remain in $\mathbb{F}_q$). Let $\#E(\mathbb{F}_q) = n$ and write n as $n = p + 1 - t$; t is called the trace of the Frobenius endomorphism. By Hasse's theorem, $|t| \leqslant 2\sqrt{q}$.

An (additive) Abelian group structure is defined on E by the well known chord-and-tangent method [91]. The order of a point $P \in E$ is the least nonzero integer r such that $[r]P = \infty$, where $[r]P$ is the sum of r terms equal to P. The order r of a point divides the curve order n. For a given integer r, the set of all points $P \in E$ such that $[r]P = \infty$ is denoted $E[r]$. We say that $E[r]$ has *embedding degree* k if $r \mid q^k - 1$ and $r \nmid q^s - 1$ for any $0 < s < k$.

The *complex multiplication* (CM) method [37] constructs an elliptic curve with a given number of points n over a given finite field $\mathbb{F}_q$ as long as $n = q+1-t$ as required by the Hasse bound, and the norm equation $DV^2 = 4q - t^2$ can be solved for "small" values of the discriminant D, from which the j-invariant of the curve (which is a function of the coefficients of the curve equation) can be computed, and the curve equation is finally given by $y^2 = x^3 + b$ (for certain values of b) when $j = 0$, by $y^2 = x^3 + ax$ (for certain values of a) when $j = 1728$, and by $y^2 = x^3 - 3cx + 2c$ with $c := j/(j - 1728)$ when $j \notin \{0, 1728\}$.

A *divisor* is a finite formal sum $\mathcal{A} = \sum_P a_P(P)$ of points on the curve $E(\mathbb{F}_{q^k})$. An Abelian group structure is defined on the set of divisors by the addition of corresponding coefficients in their formal sums; in particular, $n\mathcal{A} = \sum_P (n\, a_P)(P)$. The *degree* of a divisor $\mathcal{A}$ is the sum $\deg(\mathcal{A}) = \sum_P a_P$. Let $f : E(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}$ be a function on the curve. We define $f(\mathcal{A}) \equiv \prod_P f(P)^{a_P}$. Let $\mathrm{ord}_P(f)$ denote the multiplicity of the zero or pole of f at P (if f has no zero or pole at P, then $\mathrm{ord}_P(f) = 0$). The divisor of f is $(f) := \sum_P \mathrm{ord}_P(f)(P)$. A divisor $\mathcal{A}$ is called *principal* if $\mathcal{A} = (f)$ for some function (f). A divisor $\mathcal{A}$ is principal if and only if $\deg(\mathcal{A}) = 0$ and $\sum_P a_P P = \infty$ [65, theorem 2.25]. Two divisors $\mathcal{A}$ and $\mathcal{B}$ are *equivalent*, $\mathcal{A} \sim \mathcal{B}$, if their difference $\mathcal{A} - \mathcal{B}$ is a principal divisor. Let $P \in E(\mathbb{F}_q)[r]$ where r is coprime to q, and let $\mathcal{A}_P$ be a divisor equivalent to $(P) - (\infty)$; under these circumstances the divisor $r\mathcal{A}_P$ is principal, and hence there is a function f_P such that $(f_P) = r\mathcal{A}_P = r(P) - r(\infty)$.

Given three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of the same prime order n, a *pairing* is a feasibly computable, non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are commonly (in the so-called *Type III* pairing setting) determined by the eigenspaces of the Frobenius endomorphism ϕ_q on some elliptic curve $E/\mathbb{F}_q$ of embedding degree $k > 1$. More precisely, $\mathbb{G}_1$ is taken to be the 1-eigenspace $E[n] \cap \ker(\phi_q - [1]) = E(\mathbb{F}_q)[n]$. The group $\mathbb{G}_2$ is usually taken to be the preimage $E'(\mathbb{F}_{q^g})[n]$ of the q-eigenspace $E[n] \cap \ker(\phi_q - [q]) \subseteq E(\mathbb{F}_{q^k})[n]$ under a twisting isomorphism $\psi : E' \to E$, $(x, y) \mapsto (\mu^2 x, \mu^3 y)$ for some $\mu \in \mathbb{F}_{q^k}^*$. In particular, $g = k/d$ where the curve $E'/\mathbb{F}_{q^g}$ is the unique twist of E with largest possible twist degree $d \mid k$ for which n divides $\#E'(\mathbb{F}_{q^g})$ (see [55] for details). This means that g is as small as possible.

A Miller function $f_{i,P}$ is a function with divisor $(f_{i,P}) = i(P) - ([i]P) - (i-1)(\infty)$. Miller functions are at the root of most if not all pairings proposed for cryptographic purposes, which in turn induce efficient algorithms derived from Miller's algorithm [67, 68]. A Miller function satisfies $f_{a+b,P}(Q) = f_{a,P}(Q) \cdot f_{b,P}(Q) \cdot g_{[a]P,[b]P}(Q)/g_{[a+b]P}(Q)$ up to a constant nonzero factor in $\mathbb{F}_q$, for all $a, b \in \mathbb{Z}$, where the so-called line functions $g_{[a]P,[b]P}$ and $g_{[a+b]P}$ satisfy $(g_{[a]P,[b]P}) = ([a]P) + ([b]P) + (-[a+b]P) - 3(\infty)$, $(g_{[a+b]P}) = ([a+b]P) + (-[a+b]P) - 2(\infty)$. The advantage of Miller functions with respect to elliptic curve arithmetic is now clear, since with these relations the line functions, and hence the Miller functions themselves, can be efficiently computed as a side result during the computation of $[n]P$ by means of the usual chord-and-tangent method.

2.1 Protocols and Assumptions

As an illustration of the enormous flexibility that pairings bring to the construction of cryptographic protocols, we present a (necessarily incomplete) list of known schemes according to their overall category.

Foremost among pairing-based schemes are the identity-based cryptosystems. These include plain encryption [17], digital signatures [24, 83], (authenticated) key agreement [25], chameleon hashing [27], and hierarchical extensions thereof with or without random oracles [51, 22].

Other pairing-based schemes are not identity-based but feature special functionalities like secret handshakes [5], short/aggregate/verifiably encrypted/group/ring/blind signatures [19, 20, 26, 97, 98] and signcryption [9, 21, 61].

Together with the abundance of protocols came a matching abundance of security assumptions, often tailored to the nature of each particular protocol although some assumptions found a more general use and became classical. Some of the most popular and useful security assumptions occurring in security proofs of pairing-based protocols are the following, with groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order n in multiplicative notation (and $\mathbb{G}$ denotes either group):

- q-Strong Diffie-Hellman (q-SDH) [16] and many related assumptions (like the Inverse Computational Diffie-Hellman (Inv-CDH), the Square Computational Diffie-Hellman (Squ-CDH), the Bilinear Inverse Diffie-Hellman (BIDH), and the Bilinear Square Diffie-Hellman (BSDH) assumptions [98]): *Given a* $(\mathsf{q}+2)$*-tuple* $(g_1, g_2, g_2^x, \ldots, g_2^{x^{\mathsf{q}}}) \in \mathbb{G}_1 \times \mathbb{G}_2^{\mathsf{q}+1}$ *as input, compute a pair* $(c, g_1^{1/(x+c)}) \in \mathbb{Z}/n\mathbb{Z} \times \mathbb{G}_1$.
- Decision Bilinear Diffie-Hellman (DBDH) [18] and related assumptions (like the k-BDH assumption [14]): *Given generators* g_1 *and* g_2 *of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *respectively, and given* g_1^a, g_1^b, g_1^c, g_2^a, g_2^b, g_2^c, $e(g_1, g_2)^z$ *determine whether* $e(g_1, g_2)^{abc} = e(g_1, g_2)^z$.
- Gap Diffie-Hellman (GDH) assumption [77]: *Given* $(g, g^a, g^b) \in \mathbb{G}^3$ *for a group* $\mathbb{G}$ *equipped with an oracle for deciding whether* $g^{ab} = g^c$ *for any given* $g^c \in \mathbb{G}$, *find* g^{ab}.

– $(k+1)$ Exponent Function meta-assumption: *Given a function* $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ *and a sequence* $(g, g^a, g^{f(h_1+a)}, \ldots, g^{f(h_k+a)}) \in \mathbb{G}_1^{k+2}$ *for some* a, $h_1, \ldots, h_k \in \mathbb{Z}/n\mathbb{Z}$, *compute* $g^{f(h+a)}$ *for some* $h \notin \{h_1, \ldots, h_k\}$.

The last of these is actually a meta-assumption, since it is parameterized by a function f on the exponents. This meta-assumption includes the Collusion attack with k traitors (k-CAA) assumption [70], where $f(x) := 1/x$, and the $(k+1)$ Square Roots ($(k+1)$-SR) assumption [96], where $f(x) := \sqrt{x}$, among others. Of course, not all choices of f may lead to a consistent security assumption (for instance, the constant function is certainly a bad choice), so the instantiation of this meta-assumption must be done in a case-by-case basis.

Also, not all of these assumptions are entirely satisfactory from the point of view of their relation to the computational complexity of the more fundamental discrete logarithm problem. In particular, the Cheon attack [28, 29] showed that, contrary to most discrete-logarithm style assumptions, which usually claim a practical security level of 2^λ for 2λ-bit keys due to e.g. the Pollard-ρ attack [81], the q-SDH assumption may need 3λ-bit keys to attain that security level, according to the choice of q.

3 Curves and Algorithms

3.1 Supersingular curves

Early proposals to obtain efficient pairings invoked the adoption of supersingular curves [49, 82, 40], which led to the highly efficient concept of η pairings [7] over fields of small characteristic. This setting enables the so called Type I pairings, which are defined with both arguments from the same group [50] and facilitates the description of many protocols and the construction of formal security proofs. Unfortunately, recent developments bring that approach into question, since discrete logarithms in the multiplicative groups of the associated extension fields have proven far easier to compute than anticipated [6].

Certain ordinary curves, on the other hand, are not known to be susceptible to that line of attack, and also yield very efficient algorithms, as we will see next.

3.2 Generic constructions

Generic construction methods enable choosing the embedding degree at will, limited only by efficiency requirements. Two such constructions are known:

– The Cocks-Pinch construction [32] enables the construction of elliptic curves over $\mathbb{F}_q$ containing a pairing-friendly group of order n with $\lg(q)/\lg(n) \approx 2$.
– The Dupont-Enge-Morain strategy [39] is similarly generic in the sense of its embedding degree flexibility by maximizing the trace of the Frobenius endomorphism. Like the Cocks-Pinch method, it only attains $\lg(q)/\lg(n) \approx 2$.

.

Because the smallest attainable ratio $\lg(q)/\lg(n)$ is relatively large, these methods do not yield curves of prime order, which are necessary for certain applications like short signatures, and also tend to improve the overall processing efficiency.

3.3 Sparse families of curves

Certain families of curves may be obtained by parameterizing the norm equation $4q - t^2 = 4hn - (t-2)^2 = DV^2$ with polynomials $q(u)$, $t(u)$, $h(u)$, $n(u)$, then choosing $t(u)$ and $h(u)$ according to some criteria (for instance, setting $h(u)$ to be some small constant polynomial yields near-prime order curves), and directly finding integer solutions (in u and V) to the result. In practice this involves a clever mapping of the norm equation into a Pell-like equation, whose solutions lead to actual curve equations via complex multiplication (CM).

The only drawback they present is the relative rarity of suitable curves (the only embedding degrees that are known to yield solutions are $k \in \{3, 4, 6, 10\}$, and the size of the integer solutions u grows exponentially), especially those with prime order. Historically, sparse families are divided into Miyaji-Nakabayashi-Takano (MNT) curves and Freeman curves.

MNT curves were the first publicly known construction of ordinary pairing-friendly curves [71]. Given their limited range of admissible embedding degrees (namely, $k \in \{3, 4, 6\}$), the apparent finiteness of MNT curves of prime order [63, 58, 92], and efficiency considerations (see e.g. [44]), MNT curves are less useful for higher security levels (say, from about 2^{112} onward).

Freeman curves [43], with embedding degree $k = 10$, are far rarer and suffer more acutely from the fact that the nonexistence of a twist of degree higher than quadratic forces its $\mathbb{G}_2$ group to be defined over $\mathbb{F}_{q^5}$. Besides, this quintic extension cannot be constructed using a binomial representation.

3.4 Complete families of curves

Instead of trying to solve the partially parameterized norm equation $4h(u)n(u) - (t(u)-2)^2 = DV^2$ for u and V directly as for the sparse families of curves, one can also parameterize $V = V(u)$ as well. Solutions may exist if the parameters can be further constrained, which is usually done by considering the properties of the number field $\mathbb{Q}[u]/n(u)$, specifically by requiring that it contains a k-th root of unity where k is the desired embedding degree. Choosing $n(u)$ to be a cyclotomic polynomial $\Phi_\ell(u)$ with $k \mid \ell$ yields the suitably named cyclotomic family of curves [10, 11, 23, 44], which enable a reasonably small ratio $\rho := \lg(q)/\lg(n)$ (e.g. $\rho = (k+1)/(k-1)$ for prime $k \equiv 3 \pmod 4$).

Yet, there is one other family of curves that attain $\rho \approx 1$, namely, the Barreto-Naehrig (BN) curves [12]. BN curves arguably constitute one of the most versatile classes of pairing-friendly elliptic curves. A BN curve is an elliptic curve E_u :

$y^2 = x^3 + b$ defined over a finite prime[4] field $\mathbb{F}_p$ of (typically prime) order n, where p and n are given by $p = p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ and $n = n(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$ (hence $t = t(u) = 6u^2 + 1$) for $u \in \mathbb{Z}$. One can check by straightforward inspection that $\Phi_{12}(t(u) - 1) = n(u)n(-u)$, hence $\Phi_{12}(p(u)) \equiv \Phi_{12}(t(u) - 1) \equiv 0 \pmod{n(u)}$, so the group of order $n(u)$ has embedding degree $k = 12$.

BN curves also have j-invariant 0, so there is no need to resort explicitly to the CM curve construction method: all one has to do is choose an integer u of suitable size such that p and n as given by the above polynomials are prime. To find a corresponding curve, one chooses $b \in \mathbb{F}_p$ among the six possible classes so that the curve $E : y^2 = x^3 + b$ has order n.

Furthermore, BN curves admit a sextic twist ($d = 6$), so that one can set $\mathbb{G}_2 = E'(\mathbb{F}_{p^2})[n]$. This twist $E'/\mathbb{F}_{p^2}$ may be selected by finding a non-square and non-cube $\xi \in \mathbb{F}_{p^2}$ and then checking via scalar multiplication whether the curve $E' : y^2 = x^3 + b'$ given by $b' = b/\xi$ or by $b' = b/\xi^5$ has order divisible by n. However, construction methods are known that dispense with such procedure, yielding the correct curve and its twist directly [80]. For convenience, following [85] we call the twist $E' : y^2 = x^3 + b/\xi$ a D-type twist, and we call the twist $E' : y^2 = x^3 + b\xi$ an M-type twist.

3.5 Holistic families

Early works targeting specifically curves that have some efficiency advantage have focused on only one or a few implementation aspects, notably the pairing computation itself [13, 38, 90, 15].

More modern approaches tend to consider most if not all efficiency aspects that arise in pairing-based schemes [34, 36, 80]. This means that curves of those families tend to support not only fast pairing computation, but efficient finite field arithmetic for all fields involved, curve construction, generator construction for both $\mathbb{G}_1$ and $\mathbb{G}_2$, multiplication by a scalar in both $\mathbb{G}_1$ and $\mathbb{G}_2$, point sampling, hashing to the curve [42], and potentially other operations as well.

Curiously enough, there is not a great deal of diversity among the most promising such families, which comprise essentially only BN curves, BLS curves [10], and KSS curves [57].

3.6 Efficient algorithms

Ordinary curves with small embedding degree also come equipped with efficient pairing algorithms, which tend to be variants of the Tate pairing [8, 48, 55, 60, 76] (although some fall back to the Weil pairing while remaining fairly efficient [94]). In particular, one now knows concrete practical limits to how efficient a pairing can be, in the form of the so-called optimal pairings [93].

[4] Although there is no theoretical reason not to choose p to be a higher prime power, in practice such parameters are exceedingly rare and anyway unnecessary, so usually p is taken to be simply a prime.

As we pointed out, Miller functions are essential to the definition of most cryptographic pairings. Although all pairings can be defined individually in formal terms, it is perhaps more instructive to give the following constructive definitions, assuming an underlying curve $E/\mathbb{F}_q$ containing a group $E(\mathbb{F}_q)[n]$ of prime order n with embedding degree k and letting $z := (q^k - 1)/n$:

- Weil pairing: $w(P,Q) := (-1)^n f_{n,P}(Q)/f_{n,Q}(P)$.
- Tate pairing: $\tau(P,Q) := f_{n,P}(Q)^z$.
- Eta pairing [7] (called the twisted Ate pairing when defined over an ordinary curve): $\eta(P,Q) := f_{\lambda,P}(Q)^z$ where $\lambda^d \equiv 1 \pmod{n}$.
- Ate pairing [55]: $a(P,Q) := f_{t-1,Q}(P)^z$, where t is the trace of the Frobenius.
- Optimized Ate and twisted Ate pairings [64]: $a_c(P,Q) := f_{(t-1)^c \bmod n,Q}(P)^z$, $\eta_c(P,Q) := f_{\lambda^c \bmod n,P}(Q)^z$, for some $0 < c < k$.
- Optimal Ate pairing [93]: $a_{\text{opt}}(P,Q) := f_{\ell,Q}(P)^z$ for a certain ℓ such that $\lg \ell \approx (\lg n)/\varphi(k)$.

Optimal pairings achieve the shortest loop length among all of these pairings. To obtain unique values, most of these pairings (the Weil pairing is an exception) are reduced via the final exponentiation by z. The very computation of z is the subject of research *per se* [89]. In particular, for a BN curve with parameter u there exists an optimal Ate pairing with loop length $\ell = |6u + 2|$.

A clear trend in recent works has been to attain exceptional performance gains by limiting the allowed curves to a certain subset, sometimes to a single curve at a useful security level [75, 15, 80, 4]. In the next section, we discuss aspects pertaining such implementations.

4 Implementation aspects

The optimal Ate pairing on BN curves has been the focus of intense implementation research in the last few years. Most remarkably, beginning in 2008, a series of works improved, each one on top of the preceding one, the practical performance on Intel 64-bit platforms [54, 75, 15]. This effort reached its pinnacle in 2011, when Aranha et al. [4] reported an implementation running in about half a millisecond (see also [62]). Since then, performance of efficient software implementations has mostly stabilized, but some aspects of pairing computation continously improved through the availability of new techniques [47], processor architecture revisions and instruction set refinements [79]. In this section, we revisit the problem of efficient pairing computation working on top of the implementation presented in [4], to explore these latest advances and provide new performance figures. Our updated implementation achieves high performance on a variety of modern 64-bit computing platforms, including both relatively old processors and latest microarchitectures.

4.1 Pairing algorithm

The BN family of curves is ideal from an implementation point of view. Having embedding degree $k = 12$, it is perfectly suited to the 128-bit security level and

a competitive candidate at the 192-bit security level for protocols involving a small number of pairing computations [2]. Additionally, the size of the family facilitates generation [80] and supports many different parameter choices, allowing for customization of software implementations to radically different computing architectures [52, 53, 4]. The optimal Ate pairing construction applied to general BN curves further provides a rather simple formulation among the potential candidates [60, 76]:

$$\begin{aligned} a_{\text{opt}} : \mathbb{G}_2 \times \mathbb{G}_1 &\to \mathbb{G}_T \\ (Q, P) &\mapsto (f_{\ell,Q}(P) \cdot g_{[\ell]Q,\phi_p(Q)}(P) \cdot g_{[\ell]Q+\phi_p(Q),-\phi_p^2(Q)}(P))^{\frac{p^{12}-1}{n}}, \end{aligned}$$

with $\ell = 6u + 2$, map ϕ_p and groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as previously defined; and an especially efficient modification of Miller's Algorithm for accumulating all the required line evaluations in the Miller variable f (Algorithm 1).

The extension field arithmetic involving f is in fact the main building block of the pairing computation, including Miller's algorithm and final exponentiation. Hence, its efficient implementation is crucial. To that end, it has been recommended to implement the extension field through a tower of extensions built with appropriate choices of irreducible polynomials [38, 54, 15, 80]:

$$\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2 - \beta), \text{ with } \beta \text{ a non-square}, \tag{1}$$
$$\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[s]/(s^2 - \xi), \text{ with } \xi \text{ a non-square}, \tag{2}$$
$$\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi), \text{ with } \xi \text{ a non-cube}, \tag{3}$$
$$\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[t]/(t^3 - s) \tag{4}$$
$$\text{or } \mathbb{F}_{p^6}[w]/(w^2 - v) \tag{5}$$
$$\text{or } \mathbb{F}_{p^2}[w]/(w^6 - \xi), \text{ with } \xi \text{ a non-square and non-cube}. \tag{6}$$

Note that ξ is the same non-residue used to define the twist equations in Section 3.4 and that converting from one towering scheme to another is possible by simply reordering coefficients. By allowing intermediate values to grow to double precision and choosing p to be a prime number slightly smaller than a multiple of the processor word, lazy reduction can be efficiently employed in all levels of the towering arithmetic [4]. A remarkably efficient set of parameters arising from the curve choice $E(\mathbb{F}_p) : y^2 = x^3 + 2$, with $p \equiv 3 \pmod 4$, is $\beta = -1$, $\xi = (1 + i)$ [80], simultaneously optimizing finite field and curve arithmetic.

4.2 Field arithmetic

Prime fields involved in pairing computation in the asymmetric setting are commonly represented with dense moduli, resulting from the parameterized curve constructions. While the particular structure of the prime modulus has been successfully exploited for performance optimization in both software [75] and hardware [41], current software implementations rely on the standard Montgomery reduction [72] and state-of-the-art hardware implementations on the parallelization capabilities of the Residue Number System [30].

Algorithm 1 Optimal Ate pairing on general BN curves [4].

Input: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, \ell = |6u+2| = \sum_{i=0}^{\log_2(\ell)} \ell_i 2^i$
Output: $a_{\text{opt}}(Q,P)$

$d \leftarrow g_{Q,Q}(P)$, $T \leftarrow 2Q, e \leftarrow 1$
if $\ell_{\lfloor \log_2(\ell) \rfloor - 1} = 1$ **then** $e \leftarrow g_{T,Q}(P), T \leftarrow T + Q$
$f \leftarrow d \cdot e$
for $i = \lfloor \log_2(\ell) \rfloor - 2$ **downto** 0 **do**
    $f \leftarrow f^2 \cdot g_{T,T}(P), T \leftarrow 2T$
    if $\ell_i = 1$ **then** $f \leftarrow f \cdot g_{T,Q}(P), T \leftarrow T + Q$
end for
$Q_1 \leftarrow \phi_p(Q), Q_2 \leftarrow \phi_p^2(Q)$
if $u < 0$ **then** $T \leftarrow -T, f \leftarrow f^{p^6}$
$d \leftarrow g_{T,Q_1}(P), T \leftarrow T + Q_1, e \leftarrow g_{T,-Q_2}(P), T \leftarrow T - Q_2$, $f \leftarrow f \cdot (d \cdot e)$
$f \leftarrow f^{(p^6-1)(p^2+1)(p^4-p^2+1)/n}$
return f

Arithmetic in the base field is usually implemented in carefully scheduled Assembly code, but the small number of words required to represent a 256-bit prime field element in a 64-bit processor encourages the use of Assembly directly in the quadratic extension field, to avoid penalties related to frequent function calls [15]. Multiplication and reduction in $\mathbb{F}_p$ are implemented through a Comba strategy [33], but a Schoolbook approach is favored in recent Intel processors, due to the availability of the carry-preserving multiplication instruction `mulx`, allowing delayed handling of carries [79]. Future processors will allow similar speedups on the Comba-based multiplication and Montgomery reduction routines by carry-preserving addition instructions [78].

Divide-and-conquer approaches are used only for multiplication in $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$, because Karatsuba is typically more efficient over extension fields, since additions are relatively inexpensive in comparison with multiplication. The full details of the formulas that we use in our implementation of extension field arithmetic can be found in [4], including the opportunities for reducing the number of Montgomery reductions via lazy reduction. The case of squaring is relatively more complex. We use the complex squaring in $\mathbb{F}_{p^2}$ and, for $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$, we employ the faster Chung-Hasan asymmetric SQR3 formula [31]. The sparseness of the line functions motivates the implementation of specialized multiplication routines for accumulating the line function into the Miller variable f (*sparse* multiplication) or for multiplying line functions together (*sparser* multiplication). For sparse multiplication over $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$, we use the formulas proposed by Grewal et al. (see Algorithms 5 and 6 in [53]). Faster formulas for sparser multiplication can be trivially obtained by adapting the sparse multiplication formula to remove operations involving the missing subfield elements.

In the following, we closely follow notation for operation costs from [4]. Let m, s, a, i denote the cost of multiplication, squaring, addition and inversion in $\mathbb{F}_p$, respectively; $\tilde{m}, \tilde{s}, \tilde{a}, \tilde{\imath}$ denote the cost of multiplication, squaring, addition and

inversion in $\mathbb{F}_{p^2}$, respectively; m_u, s_u, r denote the cost of unreduced multiplication and squaring producing double-precision results, and modular reduction of double-precision integers, respectively; $\tilde{m}_u, \tilde{s}_u, \tilde{r}$ denote the cost of unreduced multiplication and squaring, and modular reduction of double-precision elements in $\mathbb{F}_{p^2}$, respectively. To simplify the operation count, we consider the cost of field subtraction, negation and division by two equivalent to that of field addition. Also, one double-precision addition is considered equivalent to the cost of two single-precision additions.

4.3 Curve arithmetic

Pairings can be computed over elliptic curves represented in any coordinate system, but popular choices have been homogeneous projective and affine coordinates, depending on the ratio between inversion and multiplication. Jacobian coordinates were initially explored in a few implementations [75, 15], but ended superseded by homogeneous coordinates because of their superior efficiency [35]. Point doublings and their corresponding line evaluations usually dominate the cost of the Miller loop, since efficient parameters tend to minimize the Hamming weight of the Miller variable ℓ and the resulting number of points additions. Below, we review and slightly refine the best formulas available for the curve arithmetic involved in pairing computation on affine and homogeneous projective coordinates.

Affine coordinates. The choice of affine coordinates has proven more useful at higher security levels and embedding degrees, due to the action of the norm map on simplifying the computation of inverses at higher extensions [86, 59]. The main advantages of affine coordinates are the simplicity of implementation and format of the line functions, allowing faster accumulation inside the Miller loop if the additional sparsity is exploited. If $T = (x_1, y_1)$ is a point in $E'(\mathbb{F}_{p^2})$, one can compute the point $2T := T + T$ with the following formula [53]:

$$\lambda = \frac{3x_1^2}{2y_1}, \;\; x_3 = \lambda^2 - 2x_1, \;\; y_3 = (\lambda x_1 - y_1) - \lambda x_3. \tag{7}$$

When E' is a D-type twist given by the twisting isomorphism ψ, the tangent line evaluated at $P = (x_P, y_P)$ has the format $g_{2\psi(T)}(P) = y_P - \lambda x_P w + (\lambda x_1 - y_1)w^3$ according to the tower representation given by Equation (6). This function can be evaluated at a cost of $3\tilde{m} + 2\tilde{s} + 7\tilde{a} + \tilde{i} + 2m$ with the precomputation cost of $1a$ to compute $\overline{x}_P = -x_P$ [53]. By performing more precomputation as $y'_P = 1/y_P$ and $x'_P = \overline{x}_P/y_P$, we can simplify the tangent line further:

$$y'_P \cdot g_{2\psi(T)}(P) = 1 + \lambda x'_P w + y'_P(\lambda x_1 - y_1)w^3.$$

Since the final exponentiation eliminates any subfield element multiplying the pairing value, this modification does not change the pairing result. Computing

the simpler line function now requires $3\tilde{m} + 2\tilde{s} + 7\tilde{a} + \tilde{\imath} + 4m$ with an additional precomputation cost of $(i + m)$:

$$A = \frac{1}{2y_1},\ \ B = 3x_1^2,\ \ C = AB,\ \ D = 2x_1,\ \ x_3 = C^2 - D,$$
$$E = Cx_1 - y_1,\ \ y_3 = E - Cx_3,\ \ F = Cx'_P,\ \ G = Ey'_P,$$
$$y'_P \cdot g_{2\psi(T)}(P) = 1 + Fw + Gw^3.$$

This clearly does not save any operations compared to Equation (7) and increases the cost by $2m$. However, the simpler format allows the faster accumulation $f^2 \cdot g_{2\psi(T)}(P) = (f_0 + f_1 w)(1 + g_1 w)$, where $f_0, f_1, g_1 \in \mathbb{F}_{p^6}$, by saving $6m$ corresponding to the multiplication between y_P and each subfield element of f_0. The performance trade-off compared to [53] is thus $4m$ per Miller doubling step.

When different points $T = (x_1, y_1)$ and $Q = (x_2, y_2)$ are considered, the point $T + Q$ can be computed with the following formula:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},\ \ x_3 = \lambda^2 - x_2 - x_1,\ \ y_3 = \lambda(x_1 - x_3) - y_1. \tag{8}$$

Applying the same trick described above gives the same performance trade-off, with a cost of $3\tilde{m} + \tilde{s} + 6\tilde{a} + \tilde{\imath} + 4m$ [53]:

$$A = \frac{1}{x_2 - x_1},\ \ B = y_2 - y_1,\ \ C = AB,\ \ D = x_1 + x_2,\ \ x_3 = C^2 - D,$$
$$E = Cx_1 - y_1,\ \ y_3 = E - Cx_3,\ \ F = Cx'_P,\ \ G = Ey'_P,$$
$$y'_P \cdot g_{\psi(T),\psi(Q)}(P) = 1 + Fw + Gw^3.$$

The technique can be further employed in M-type twists, conserving their equivalent performance to D-type twists [53], with some slight changes in the formula format and accumulation multiplier. A generalization for other pairing-friendly curves with degree-d twists and even embedding degree k would provide a performance trade-off of $(k/2 - k/d)$ multiplications per step in Miller's Algorithm. The same idea was independently proposed and slightly improved in [73].

Homogeneous Projective coordinates. The choice of projective coordinates has proven especially advantageous at the 128-bit security level for single pairing computation, due to the typically large inversion/multiplication ratio in this setting. If $T = (X_1, Y_1, Z_1) \in E'(\mathbb{F}_{p^2})$ is a point in homogeneous coordinates, one can compute the point $2T = (X_3, Y_3, Z_3)$ with the following formula [4]:

$$\begin{aligned} X_3 &= \frac{X_1 Y_1}{2}(Y_1^2 - 9b'Z_1^2), \\ Y_3 &= \left[\frac{1}{2}(Y_1^2 + 9b'Z_1^2)\right]^2 - 27b'^2 Z_1^4,\ \ Z_3 = 2Y_1^3 Z_1. \end{aligned} \tag{9}$$

The twisting point P can be represented by $(x_P w, y_P)$. When E' is a D-type twist given by the twisting isomorphism ψ, the tangent line evaluated at $P = (x_P, y_P)$ can be computed with the following formula [53]:

$$g_{2\psi(T)}(P) = -2YZy_P + 3X^2 x_P w + (3b'Z^2 - Y^2)w^3 \tag{10}$$

Equation (10) is basically the same line evaluation formula presented in [35] plus an efficient selection of the positioning of terms (obtained by multiplying the line evaluation by w^3), which was suggested in [53] to obtain a fast sparse multiplication in the Miller loop (in particular, the use of terms $1, w$ and w^3 [53] induces a sparse multiplication that saves $13\tilde{a}$ in comparison to the use of terms $1, v^2$ and wv in [4]). The full doubling/line function formulae in [35] costs $2\tilde{m} + 7\tilde{s} + 23\tilde{a} + 4m + m_{b'}$. Based on Equations (9) and (10), [53] reports a cost of $2\tilde{m} + 7\tilde{s} + 21\tilde{a} + 4m + m_{b'}$. We observe that the same formulae can be evaluated at a cost of only $2\tilde{m} + 7\tilde{s} + 19\tilde{a} + 4m + m_{b'}$ with the precomputation cost of $3a$ to compute $\overline{y}_P = -y_P$ and $x'_P = 3x_P$. Note that all these costs consider the computation of $X_1 \cdot Y_1$ using the equivalence $2XY = (X + Y)^2 - X^2 - Y^2$. We remark that, as in Aranha et al. [4], on x64 platforms it is more efficient to compute such term with a direct multiplication since $\tilde{m} - \tilde{s} < 3\tilde{a}$. Considering this scenario, the cost applying our precomputations is then given by $3\tilde{m} + 6\tilde{s} + 15\tilde{a} + 4m + m_{b'}$. Finally, further improvements are possible if b is cleverly selected [80]. For instance, if $b = 2$ then $b' = 2/(1 + i) = 1 - i$, which minimizes the number of additions and subtractions. Computing the simpler doubling/line function now requires $3\tilde{m} + 6\tilde{s} + 16\tilde{a} + 4m$ with the precomputation cost of $3a$ (in comparison to the computation proposed in [53], [4] and [35], we save $2\tilde{a}, 3\tilde{a}$ and $5\tilde{a}$, respectively, when $\tilde{m} - \tilde{s} < 3\tilde{a}$):

$$\begin{gathered} A = X_1 \cdot Y_1/2,\ \ B = Y_1^2,\ \ C = Z_1^2,\ \ D = 3C,\ \ E_0 = D_0 + D_1, \\ E_1 = D_1 - D_0,\ \ F = 3E,\ \ X_3 = A \cdot (B - F),\ \ G = (B + F)/2, \\ Y_3 = G^2 - 3E^2,\ \ H = (Y_1 + Z_1)^2 - (B + C),\ \ Z_3 = B \cdot H, \\ g_{2\psi(T)}(P) = H\overline{y}_P + X_1^2 x'_P w + (E - B)w^3. \end{gathered} \tag{11}$$

Similarly, if $T = (X_1, Y_1, Z_1)$ and $Q = (x_2, y_2) \in E'(\mathbb{F}_{p^2})$ are points in homogeneous and affine coordinates, respectively, one can compute the point $T + Q = (X_3, Y_3, Z_3)$ with the following formula:

$$\begin{gathered} X_3 = \lambda(\lambda^3 + Z_1\theta^2 - 2X_1\lambda^2), \\ Y_3 = \theta(3X_1\lambda^2 - \lambda^3 - Z_1\theta^2) - Y_1\lambda^3,\ \ Z_3 = Z_1\lambda^3, \end{gathered} \tag{12}$$

where $\theta = Y_1 - y_2 Z_1$ and $\lambda = X_1 - x_2 Z_1$. In the case of a D-type twist, the line evaluated at $P = (x_P, y_P)$ can be computed with the following formula [53]:

$$g_{\psi(T+Q)}(P) = -\lambda y_P - \theta x_P w + (\theta X_2 - \lambda Y_2)w^3. \tag{13}$$

Similar to the case of doubling, Equation (13) is basically the same line evaluation formula presented in [35] plus an efficient selection of the positioning of terms suggested in [53] to obtain a fast sparse multiplication inside the Miller loop. The full mixed addition/line function formulae can be evaluated at a cost of $11\tilde{m} + 2\tilde{s} + 8\tilde{a} + 4m$ with the precomputation cost of $2a$ to compute $\overline{x}_P = -x_P$

and $\overline{y}_P = -y_P$ [53]:

$$A = Y_2Z_1,\ \ B = X_2Z_1,\ \ \theta = Y_1 - A,\ \ \lambda = X_1 - B,\ \ C = \theta^2,$$
$$D = \lambda^2,\ \ E = \lambda^3,\ \ F = Z_1C,\ \ G = X_1D,\ \ H = E + F - 2G,$$
$$X_3 = \lambda H,\ \ I = Y_1E,\ \ Y_3 = \theta(G - H) - I,\ \ Z_3 = Z_1E,\ \ J = \theta X_2 - \lambda Y_2,$$
$$g_{2\psi(T)}(P) = \lambda\overline{y}_P + \theta\bar{x}_P w + Jw^3.$$

In the case of an M-type twist, the line function evaluated at $\psi(P) = (x_P w^2, y_P w^3)$ can be computed with the same sequence of operations shown above.

4.4 Operation count

Table 1 presents a detailed operation count for each operation relevant in the computation of a pairing over a BN curve, considering all the improvements described in the previous section. Using these partial numbers, we obtain an operation count for the full pairing computation on a fixed BN curve.

Miller loop. Sophisticated pairing-based protocols may impose additional restrictions on the parameter choice along with some performance penalty, for example requiring the cofactor of the $\mathbb{G}_T$ group to be a large prime number [87]. For efficiency and a fair comparison with related works, we adopt the parameters β, ξ, $b = 2$, $u = -(2^{62} + 2^{55} + 1)$ from [80]. For this set of parameters, the Miller loop in Algorithm 1 and the final line evaluations execute some amount of precomputation for accelerating the curve arithmetic formulas, 64 points doublings with line evaluations and 6 point additions with line evaluations; a single p-power Frobenius, a single p^2-power Frobenius and 2 negations in $E'(\mathbb{F}_{p^2})$; and 66 sparse accumulations in the Miller variable, 2 sparser multiplications, 1 multiplication, 1 conjugation and 63 squarings in $\mathbb{F}_{p^{12}}$. The corresponding costs in affine and homogeneous projective coordinates are, respectively:

$$\begin{aligned}
\text{MLA} &= (i + m + a) + 64 \cdot (3\tilde{m} + 2\tilde{s} + 7\tilde{a} + \tilde{i} + 4m) \\
&\quad + 6 \cdot (3\tilde{m} + \tilde{s} + 6\tilde{a} + \tilde{i} + 4m) + 2\tilde{m} + 2a + 2m + 2\tilde{a} \\
&\quad + 66 \cdot (10\tilde{m}_u + 6\tilde{r} + 31\tilde{a}) + 2 \cdot (5\tilde{m}_u + 3\tilde{r} + 13\tilde{a}) \\
&\quad + 3\tilde{a} + (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) + 63 \cdot (3\tilde{m}_u + 12\tilde{s}_u + 6\tilde{r} + 93\tilde{a}) \\
&= 1089\tilde{m}_u + 890\tilde{s}_u + 1132\tilde{r} + 8530\tilde{a} + 70\tilde{i} + i + 283m + 3a.
\end{aligned}$$

$$\begin{aligned}
\text{MLP} &= (4a) + 64 \cdot (3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 19\tilde{a} + 4m) + \\
&\quad + 6 \cdot (11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 10\tilde{a} + 4m) + 2\tilde{m} + 2a + 2m + 2\tilde{a} \\
&\quad + 66 \cdot (13\tilde{m}_u + 6\tilde{r} + 48\tilde{a}) + 2 \cdot (6\tilde{m}_u + 5\tilde{r} + 22\tilde{a}) \\
&\quad + 3\tilde{a} + (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) + 63 \cdot (3\tilde{m}_u + 12\tilde{s}_u + 6\tilde{r} + 93\tilde{a}) \\
&= 1337\tilde{m}_u + 1152\tilde{s}_u + 1388\tilde{r} + 10462\tilde{a} + 282m + 6a.
\end{aligned}$$

Table 1. Computational cost for arithmetic required by Miller's Algorithm.

$E'(\mathbb{F}_{p^2})$-Arithmetic	Operation Count
Precomp. (Affine)	$i + m + a$
Precomp. (Proj)	$4a$
Dbl./Eval. (Affine)	$3\tilde{m} + 2\tilde{s} + 7\tilde{a} + \tilde{\imath} + 4m$
Add./Eval. (Affine)	$3\tilde{m} + \tilde{s} + 6\tilde{a} + \tilde{\imath} + 4m$
Dbl./Eval. (Proj)	$3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 19\tilde{a} + 4m$
Add./Eval. (Proj)	$11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 10\tilde{a} + 4m$
p-power Frobenius	$2\tilde{m} + 2a$
p^2-power Frobenius	$2m + \tilde{a}$
Negation	$\tilde{a}$
$\mathbb{F}_{p^2}$-Arithmetic	Operation Count
Add./Sub./Neg.	$\tilde{a} = 2a$
Conjugation	a
Multiplication	$\tilde{m} = \tilde{m}_u + \tilde{r} = 3m_u + 2r + 8a$
Squaring	$\tilde{s} = \tilde{s}_u + \tilde{r} = 2m_u + 2r + 3a$
Multiplication by β	a
Multiplication by ξ	$2a$
Inversion	$\tilde{\imath} = i + 2s_u + 2m_u + 2r + 3a$
$\mathbb{F}_{p^{12}}$-Arithmetic	Operation Count
Add./Sub.	$6\tilde{a}$
Conjugation	$3\tilde{a}$
Multiplication	$18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}$
Sparse Mult. (Affine)	$10\tilde{m}_u + 6\tilde{r} + 31\tilde{a}$
Sparser Mult. (Affine)	$5\tilde{m}_u + 3\tilde{r} + 13\tilde{a}$
Sparse Mult. (Proj)	$13\tilde{m}_u + 6\tilde{r} + 48\tilde{a}$
Sparser Mult. (Proj)	$6\tilde{m}_u + 5\tilde{r} + 22\tilde{a}$
Squaring	$3\tilde{m}_u + 12\tilde{s}_u + 6\tilde{r} + 93\tilde{a}$
Cyc. Squaring	$9\tilde{s}_u + 6\tilde{r} + 46\tilde{a}$
Comp. Squaring	$6\tilde{s}_u + 4\tilde{r} + 31\tilde{a}$
Simult. Decomp.	$9\tilde{m} + 6\tilde{s} + 22\tilde{a} + \tilde{\imath}$
p-power Frobenius	$5\tilde{m} + 6a$
p^2-power Frobenius	$10m + 2\tilde{a}$
p^3-power Frobenius	$5\tilde{m} + 2\tilde{a} + 6a$
Inversion	$23\tilde{m}_u + 11\tilde{s}_u + 16\tilde{r} + 129\tilde{a} + \tilde{\imath}$

Final exponentiation. For computing the final exponentiation, we employ the state-of-the-art approach by [47] in the context of BN curves. As initially proposed by [89], power $\frac{p^{12}-1}{r}$ is factored into the easy exponent $(p^6-1)(p^2+1)$ and the hard exponent $\frac{p^4-p^2+1}{n}$. The easy power is computed by a short sequence of multiplications, conjugations, fast applications of the Frobenius map [15] and a single inversion in $\mathbb{F}_{p^{12}}$. The hard power is computed in the cyclotomic subgroup, where additional algebraic structure allows elements to be compressed and squared consecutively in their compressed form, with decompression required only when performing multiplications [4, 88, 74].

Moreover, lattice reduction is able to obtain parameterized multiples of the hard exponent and significantly reduce the length of the addition chain involved in that exponentiation [47]. In total, the hard part of the final exponentiation requires 3 exponentiations by parameter u, 3 squarings in the cyclotomic subgroup, 10 full extension field multiplications and 3 applications of the Frobenius maps with increasing pth-powers. We refer to [4] for the cost of an exponentiation by our choice of u and compute the exact operation count of the final exponentiation:

$$\begin{aligned}\mathrm{FE} &= (23\tilde{m}_u + 11\tilde{s}_u + 16\tilde{r} + 129\tilde{a} + \tilde{i}) + 3\tilde{a} + 12 \cdot (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) \\ &\quad + 3 \cdot (45\tilde{m}_u + 378\tilde{s}_u + 275\tilde{r} + 2164\tilde{a} + \tilde{i}) + 3 \cdot (9\tilde{s}_u + 6\tilde{r} + 46\tilde{a}) \\ &\quad + (5\tilde{m} + 6a) + 2 \cdot (10m + 2\tilde{a}) + (5\tilde{m} + 2\tilde{a} + 6a) \\ &= 384\tilde{m}_u + 1172\tilde{s}_u + 941\tilde{r} + 8085\tilde{a} + 4\tilde{i} + 20m + 12a.\end{aligned}$$

4.5 Results and discussion

The combined cost for a pairing computation in homogeneous projective coordinates can then be expressed as:

$$\begin{aligned}\mathrm{MLP} + \mathrm{FE} &= 1721\tilde{m}_u + 2324\tilde{s}_u + 2329\tilde{r} + 18547\tilde{a} + 4\tilde{i} + i + 302m + 18a \\ &= 9811m_u + 4658r + 57384a + 4\tilde{i} + i + 302m + 18a \\ &= 10113m_u + 4960r + 57852a + 4\tilde{i} + i.\end{aligned}$$

A direct comparison with a previous record-setting implementation [4], considering only the number of multiplications in $\mathbb{F}_p$ generated by arithmetic in $\mathbb{F}_{p^2}$ as the performance metric, shows that our updated implementation in projective coordinates saves 3.4% of the base field multiplications. This reflects the faster final exponentiation adopted from [47] and the more efficient formulas for inversion and squaring in $\mathbb{F}_{p^{12}}$. These formulas were not the most efficient in [4] due to higher number of additions, but this additional cost is now offset by improved addition handling and faster division by 2. Now comparing the total number of multiplications with more recent implementations [95, 69], our updated implementation saves 1.9%, or 198 multiplications.

The pairing code was implemented in the C programming language, with the performance-critical code implemented in Assembly. The compiler used was GCC

version 4.7.0, with switches turned on for loop unrolling, inlining of small functions to reduce function call overhead and optimization level `-O3`. Performance experiments were executed in a broad set of 64-bit Intel-compatible platforms: older Nehalem Core i5 540M 2.53GHz and AMD Phenom II 3.0 GHz processors, and modern Sandy Bridge Xeon E31270 3.4GHz and Ivy Bridge Core i5 3570 3.4GHz processors, including a recent Haswell Core i7 4750 HQ 2.0GHz processor. All machines had automatic overclocking capabilities disabled to reduce randomness in the results. Table 2 presents the timings split in the Miller loop and final exponentiation. This is not only useful for more fine-grained comparisons, but also to allow more accurate estimates of the latency of multi-pairings or precomputed pairings. The complete implementation will be made available in the next release of the RELIC toolkit [3].

Table 2. Comparison between implementations based on affine and projective coordinates on 64-bit architectures. Timings are presented in 10^3 clock cycles and were collected as the average of 10^4 repetitions of the same operation. Target platforms are AMD Phenom II (P II) and Intel Nehalem (N), Sandy Bridge (SB), Ivy Bridge (IB), Haswell (H) with or without support to the `mulx` instruction.

	Platform					
Operation	N	P II	SB	IB	H	H+`mulx`
Affine Miller Loop	1,680	1,529	1,365	1,315	1,259	1,212
Projective Miller Loop	1,170	862	856	798	721	704
Final Exponentiation	745	557	572	537	492	473
Affine Pairing	2,425	2,086	1,937	1,852	1,751	1,685
Projective Pairing	1,915	1,419	1,428	1,335	1,213	1,177

We obtain several performance improvements in comparison with current literature. Our implementation based on projective coordinates improves results from [4] by 6% and 9% in the Nehalem and Phenom II machines, respectively. Comparing to an updated version [95] of a previous record setting implementation [15], our Sandy Bridge timings are faster by 82,000 cycles, or 5%. When independently benchmarking their available software in the Ivy Bridge machine, we observe a latency of 1,403K cycles, thus an improvement by our software of 5%. Now considering the Haswell results from the same software available at [69], we obtain a speedup of 8% without taking into account the `mulx` instruction and comparable performance when `mulx` is employed. It is also interesting to note that the use of `mulx` injects a relatively small speedup of 3%. When exploiting such an instruction, the lack of carry-preserving addition instructions in the first generation of Haswell processors makes an efficient implementation of Comba-based multiplication and Montgomery reduction difficult, favoring the use of the typically slower Schoolbook versions. We anticipate a better support for Comba variants with the upcoming addition instructions [78].

In the implementation based on affine coordinates, the state-of-the-art results at the 128-bit security level is the one described by Acar *et al.* [1]. Unfortunately, only the latency of 15,6 million cycles on a Core 2 Duo is provided for 64-bit Intel architectures. While this does not allow a direct comparison, observing the small performance improvement between the Core 2 Duo and Nehalem reported in [4] implies that our affine implementation should be around 6 times faster than [1] when executed in the same machine.

Despite being slower than our own projective version, our affine implementation is still considerably faster than some previous speed records on projective coordinates [54, 75, 15]. This hints at the possibility that affine pairings could be improved even further, contrary to the naive intuition that the affine representation is exceedingly worse than a projective approach.

5 Conclusion

Pairings are amazingly flexible tools that enable the design of innovative cryptographic protocols. Their complex implementation has been the focus of intense research since the beginning of the millennium in what became a formidable race to make it efficient and practical.

We have reviewed the theory behind pairings and covered state-of-the-art algorithms, and also presented some further optimizations to the pairing computation in affine and projective coordinates, and analyzed the performance of the most efficient algorithmic options for pairing computation over ordinary curves at the 128-bit security level. In particular, our implementations of affine and projective pairings using Barreto-Naehrig curves shows that the efficiency of these two approaches are not as contrasting as it might seem, and hints that further optimizations might be possible. Remarkably, the combination of advances in processor technology and carefully crafted algorithms brings the computation of pairings close to the one million cycle mark.

Acknowledgements

The authors would like to thank Tanja Lange for the many suggestions to improve the quality of this paper.

References

1. T. Acar, K. Lauter, M. Naehrig, and D. Shumow. Affine pairings on ARM. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography – Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 203–209. Springer, 2013.
2. D. F. Aranha, L. Fuentes-Castañeda, E. Knapp, A. Menezes, and F. Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography – Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 117–195. Springer, 2013.

3. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.
4. D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López. Faster explicit formulas for computing pairings over ordinary curves. In *Advances in Cryptology – Eurocrypt 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 48–68, Tallinn, Estonia, 2011. Springer.
5. D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H. C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy – S&P 2003*, pages 180–196, Berkeley, USA, 2003. IEEE Computer Society.
6. R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. Cryptology ePrint Archive, Report 2013/400, 2013. `http://eprint.iacr.org/2013/400`.
7. P. S. L. M. Barreto, S. D. Galbraith, C. Ó hÉigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Designs, Codes and Cryptography*, 42(3):239–271, 2007.
8. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 377–387, Santa Barbara, USA, 2002. Springer.
9. P. S. L. M. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In B. Roy, editor, *Advances in Cryptology – Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2005.
10. P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks – SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273, Amalfi, Italy, 2002. Springer.
11. P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography – SAC 2003*, Lecture Notes in Computer Science, pages 17–25, Ottawa, Canada, 2004. Springer.
12. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2006.
13. N. Benger and M. Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In M. A. Hasan and T. Helleseth, editors, *Arithmetic of Finite Fields – WAIFI 2010*, volume 6087 of *Lecture Notes in Computer Science*, pages 180–195, Istanbul, Turkey, 2010. Springer.
14. K. Benson, H. Shacham, and B. Waters. The k-BDH assumption family: Bilinear map cryptography from progressively weaker assumptions. In E. Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 310–325. Springer, 2013.
15. J.-L. Beuchat, J. E. González Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves. In M. Joye, A. Miyaji, and A. Otsuka, editors, *Pairing-Based Cryptography – Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2010.
16. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. L. Camenisch, editors, *Advances in Cryptology – Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

17. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, USA, 2001. Springer.
18. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
19. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – Eurocrypt 2003*, Lecture Notes in Computer Science, pages 416–432. Springer, 2003.
20. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, 2002. Springer.
21. X. Boyen. Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. In *Advances in Cryptology – Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399, Santa Barbara, USA, 2003. Springer.
22. X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology – Crypto 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307, Santa Barbara, USA, 2006. Springer.
23. F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. *Designs, Codes and Cryptography*, 37(1):133–141, 2005.
24. J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30, Miami, USA, 2003. Springer.
25. L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, 2007.
26. X. Chen, F. Zhang, and K. Kim. New ID-based group signature from pairings. *Journal of Electronics (China)*, 23(6):892–900, 2006.
27. X. Chen, F. Zhang, W. Susilo, H. Tian, J. Li, and K. Kim. Identity-based chameleon hash scheme without key exposure. In Ron Steinfeld and Philip Hawkes, editors, *Information Security and Privacy*, volume 6168 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2010.
28. J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology – Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2006.
29. J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, 2010.
30. R. C. C. Cheung, S. Duquesne, J. Fan, N. Guillermin, I. Verbauwhede, and G. X. Yao. FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 421–441. Springer, 2011.
31. J. Chung and M. Hasan. Asymmetric Squaring Formulae. In *18th IEEE Symposium on Computer Arithmetic – ARITH-18 2007*, pages 113–122, 2007.
32. C. Cocks and R. G. E. Pinch. Identity-based cryptosystems based on the Weil pairing. *Unpublished manuscript*, 2001.
33. P. G. Comba. Exponentiation Cryptosystems on the IBM PC. *IBM Systems Journal*, 29(4):526–538, 1990.
34. C. Costello. Particularly friendly members of family trees. Cryptology ePrint Archive, Report 2012/072, 2012. `http://eprint.iacr.org/`.

35. C. Costello, T. Lange, and M. Naehrig. Faster Pairing Computations on Curves with High-Degree Twists. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 224–242. Springer, 2010.
36. C. Costello, K. Lauter, and M. Naehrig. Attractive subfamilies of BLS curves for implementing high-security pairings. In *Progress in Cryptology – Indocrypt 2011*, volume 7107 of *Lecture Notes in Computer Science*, pages 320–342, Chennai, India, 2011. Springer.
37. R. Crandall and C. Pomerance. *Prime Numbers: a Computational Perspective*. Springer, Berlin, 2001.
38. A. J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over Barreto-Naehrig curves. In *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 197–207. Springer, 2007.
39. R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. *Journal of Cryptology*, 18(2):79–89, 2005.
40. I. Duursma and H.-S. Lee. Tate pairing implementation for hyperelliptic curves. In C.-S. Laih, editor, *Advances in Cryptology – Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 2003.
41. J. Fan, F. Vercauteren, and I. Verbauwhede. Efficient hardware implementation of $\mathbb{F}_p$-arithmetic for pairing-friendly curves. *IEEE Transactions on Computers*, 61(5):676–685, 2012.
42. P.-A. Fouque and M. Tibouchi. Indifferentiable hashing to Barreto-Naehrig curves. In *Progress in Cryptology – Latincrypt 2012*, volume 7533 of *Lecture Notes in Computer Science*, pages 1–17, Santiago, Chile, 2012. Springer.
43. D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In F. Hess, S. Pauli, and M. Pohst, editors, *Algorithmic Number Theory*, volume 4076 of *Lecture Notes in Computer Science*, pages 452–465. Springer, 2006.
44. D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.
45. G. Frey, M. Müller, and H. Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.
46. G. Frey and H. G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.
47. L. Fuentes-Castañeda, E. Knapp, and F. Rodríguez-Henríquez. Faster Hashing to $\mathbb{G}_2$. In A. Miri and S. Vaudenay, editors, *Selected Areas in Cryptography – SAC 2011*, volume 7118 of *Lecture Notes in Computer Science*, pages 412–430. Springer, 2011.
48. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory Symposium – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337, Sydney, Australia, 2002. Springer.
49. S. D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Advances in Cryptology - Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2001.
50. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
51. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *Advances in Cryptology - Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

52. C. P. L. Gouvêa and J. López. Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller. In B. K. Roy and N. Sendrier, editors, *10th International Conference on Cryptology in India – Indocrypt 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2009.
53. G. Grewal, R. Azarderakhsh, P. Longa, S. Hu, and D. Jao. Efficient Implementation of Bilinear Pairings on ARM Processors. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography – SAC 2012*, volume 7707 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 2012.
54. D. Hankerson, A. Menezes, and M. Scott. Software Implementation of Pairings. In *Identity-Based Cryptography*, chapter 12, pages 188–206. IOS Press, 2008.
55. F. Hess, N. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52:4595–4602, 2006.
56. A. Joux. A one-round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithm Number Theory Symposium – ANTS IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.
57. E. Kachisa, E. Schaefer, and M. Scott. Constructing Brezing-Weng pairing friendly elliptic curves using elements in the cyclotomic field. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer, 2008.
58. K. Karabina and E. Teske. On prime-order elliptic curves with embedding degrees $k = 3$, 4, and 6. In *Proceedings of the 8th international conference on Algorithmic number theory – ANTS-VIII*, volume 5011 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2008.
59. K. Lauter, P. Montgomery, and M. Naehrig. An analysis of affine coordinates for pairing computation. In M. Joye, A. Miyaji, and A. Otsuka, editors, *Pairing-Based Cryptography – Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2010.
60. E. Lee, H.-S. Lee, and C.-M. Park. Efficient and generalized pairing computation on abelian varieties. *IEEE Transactions on Information Theory*, 55(4):1793–1803, 2009.
61. B. Libert and J.-J. Quisquater. New identity based signcryption schemes from pairings. In *Information Theory Workshop – ITW 2003*, pages 155–158. IEEE, 2003.
62. P. Longa. *High-Speed Elliptic Curve and Pairing-Based Cryptography*. PhD thesis, University of Waterloo, April 2011.
63. F. Luca and I. E. Shparlinski. Elliptic curves with low embedding degree. *Journal of Cryptology*, 19(4):553–562, 2006.
64. S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the ate and twisted ate pairings. In S. D. Galbraith, editor, *Cryptography and Coding – IMACC 2007*, volume 4887 of *Lecture Notes in Computer Science*, pages 302–312. Springer, 2007.
65. A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, Boston, USA, 1993.
66. A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39:1639–1646, 1993.
67. V. S. Miller. Short programs for functions on curves. IBM Thomas J. Watson Research Center Report, 1986. `http://crypto.stanford.edu/miller/miller.pdf`.

68. V. S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.
69. S. Mitsunari. A fast implementation of the optimal ate pairing over BN curve on Intel Haswell processor. Cryptology ePrint Archive, Report 2013/362, 2013. `http://eprint.iacr.org/`.
70. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transactions on Fundamentals*, E85-A(2):481–484, 2002.
71. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
72. P. L. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 44(170):519–521, 1985.
73. Y. Mori, S. Akagi, Y. Nogami, and M. Shirase. Pseudo 8-Sparse Multiplication for Efficient Ate-based Pairing on Barreto-Naehrig Curve. In *Pairing-Based Cryptography – Pairing 2013*. To appear.
74. M. Naehrig, P. S. L. M. Barreto, and P. Schwabe. On compressible pairings and their computation. In S. Vaudenay, editor, *Progress in Cryptology – Africacrypt 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2008.
75. M. Naehrig, R. Niederhagen, and P. Schwabe. New software speed records for cryptographic pairings. In *Progress in Cryptology – Latincrypt 2010*, volume 6212 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2010.
76. Y. Nogami, M. Akane, Y. Sakemi, H. Kato, and Y. Morikawa. Integer variable χ-based ate pairing. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 178–191. Springer, 2008.
77. T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In K. Kim, editor, *Public Key Cryptography – PKC 2001*, pages 104–118, London, UK, 2001. Springer.
78. E. Ozturk, J. Guilford, and V. Gopal. Large Integer Squaring on Intel Architecture Processors. Intel white paper, 2013.
79. E. Ozturk, J. Guilford, V. Gopal, and W. Feghali. New Instructions Supporting Large Integer Arithmetic on Intel Architecture Processors. Intel white paper, 2012.
80. G. C. C. F. Pereira, M. A. Simplício, Jr., M. Naehrig, and P. S. L. M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, August 2011.
81. J. M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32:918–924, 1978.
82. K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In M. Yung, editor, *Advances in Cryptology- CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 336–353. Springer, 2002.
83. R. Sakai and M. Kasahara. Cryptosystems based on pairing over elliptic curve. In *Symposium on Cryptography and Information Security – SCIS 2003*, pages 8C–1, January 2003.
84. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security – SCIS 2000*, Okinawa, Japan, January 2000.
85. M. Scott. A note on twists for pairing friendly curves. `ftp://ftp.computing.dcu.ie/pub/resources/crypto/twists.pdf`, 2009.

86. M. Scott. On the efficient implementation of pairing-based protocols. In Liqun Chen, editor, *Cryptography and Coding – IMACC 2011*, volume 7089 of *Lecture Notes in Computer Science*, pages 296–308. Springer, 2011.
87. M. Scott. Unbalancing pairing-based key exchange protocols. Cryptology ePrint Archive, Report 2013/688, 2013. `http://eprint.iacr.org/2013/688`.
88. M. Scott and P. S. L. M. Barreto. Compressed pairings. In M. Franklin, editor, *Advances in Cryptology – Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156, Santa Barbara, USA, 2004. Springer.
89. M. Scott, N. Benger, M. Charlemagne, L. J. Domínguez Pérez, and E. J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 78–88. Springer Berlin Heidelberg, 2009.
90. M. Shirase. Barreto-Naehrig curve with fixed coefficient. IACR ePrint Archive, report 2010/134, 2010. `http://eprint.iacr.org/2010/134`.
91. J. H. Silverman. *The Arithmetic of Elliptic Curves*. Number 106 in Graduate Texts in Mathematics. Springer, Berlin, Germany, 1986.
92. J. Jiménez Urroz, F. Luca, and I. Shparlinski. On the number of isogeny classes of pairing-friendly elliptic curves and statistics of MNT curves. *Mathematics of Computation*, 81(278), 2012.
93. F. Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.
94. A. Weil. Sur les fonctions algébriques à corps de constantes fini. *Comptes rendus de l'Académie des sciences*, 210:592–594, 1940.
95. E. Zavattoni, L. J. Domínguez-Pérez, S. Mitsunari, A. H. Sánchez, T. Teruya, and F. Rodríguez-Henríquez. Software implementation of attribute-based encryption. `http://sandia.cs.cinvestav.mx/index.php?n=Site.CPABE`, 2013.
96. F. Zhang and X. Chen. Yet another short signatures without random oracles from bilinear pairings. IACR Cryptology ePrint Archive, report 2005/230, 2005.
97. F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In Y. Zheng, editor, *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer, 2002.
98. F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In F. Bao, R. Deng, and J. Zhou, editors, *Public Key Cryptography – PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 277–290. Springer, 2004.