Curves with fast computations in the first pairing group

Rémi Clarisse[1,2], Sylvain Duquesne[2], and Olivier Sanders[1]

[1] Orange Labs, Applied Crypto Group, Cesson-Sévigné, France
[2] Univ Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France

Abstract. Pairings are a powerful tool to build advanced cryptographic schemes. The most efficient way to instantiate a pairing scheme is through Pairing-Friendly Elliptic Curves.
Because a randomly picked elliptic curve will not support an efficient pairing (the embedding degree will usually be too large to make any computation practical), a pairing-friendly curve has to be carefully constructed. This has led to famous curves, *e.g.* Barreto-Naehrig curves.
However, the computation of the discrete logarithm problem on the finite-field side has received much interest and its complexity has recently decreased. Hence the need to propose new curves has emerged.
In this work, we give one new curve that is specifically tailored to be fast over the first pairing-group, which is well suited for several cryptographic schemes, such as group signatures and their variants (EPID, anonymous attestation, etc) or accumulators.

1 Introduction

Pairings and cryptography have a long common history. Initially used as a way to shift the discrete logarithm problem from elliptic curves to finite fields [33], it has first been used for constructive purpose by Joux [28] in 2000. Following this seminal result, pairings have been massively used in cryptography. This is due in large part to the nice features of this mathematical tool but also to its apparent simplicity. Indeed, the features of pairings can easily be abstracted so as to be used even by non-specialists. Actually, almost all pairing-based cryptographic papers (*e.g.* [8,35,5]) consider so-called "bilinear groups" as some kind of black box given by a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ (usually of prime order ℓ) along with an efficiently-computable non-degenerate bilinear map between them

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T.$$

This way, cryptographers can design their protocols without being bothered by the technicalities of the concrete implementations of pairings. The only subtlety considered in cryptographic papers is the existence of

efficiently computable morphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$, which has led to so called *type-i* pairings, for $i \in \{1, 2, 3\}$, according to the classification by Galbraith *et al.* [21]. However, type-3 pairings are now preponderant in cryptography because they are the most efficient ones [21] and because they are compatible with cryptographic assumptions, such as Decisional Diffie-Hellman in both $\mathbb{G}_1$ and $\mathbb{G}_2$, that do not hold with the other types. Actually, some recent results [14,1] cast some doubts on the real interest of type-1 and type-2 pairings for cryptography. For all these reasons, we only consider type-3 pairings in this paper.

In all cases, at some point, it becomes necessary to instantiate these bilinear groups. To date, the only secure instantiations are (to our knowledge) based on elliptic curves as the constructions based on lattices [23,17] have been proved insecure [16,15]. More specifically, $\mathbb{G}_1$ and $\mathbb{G}_2$ are usually defined as cyclic groups of prime order ℓ of some elliptic curve E over a finite field $\mathbb{F}_q$. For its part, $\mathbb{G}_T$ is the group of ℓ-th roots of unity in $\mathbb{F}_{q^k}$, where k is the order of q in $\mathbb{Z}/\ell\mathbb{Z}$, called the embedding degree of q.

It is thus important to understand that, despite being considered as similar objects by the cryptographic abstraction of bilinear groups, the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are extremely different in practice. The main difficulty when it comes to instantiate this groups is to carefully select the different parameters (essentially ℓ, q and k) in order to ensure that security holds in each group while retaining the best efficiency. By "security", we here mean the hardness of the Discrete Logarithm Problem (DLP) although cryptographic schemes usually rely on easier problems.

Actually, for a long time, the problem of selecting these parameters was thought to be rather easy. It was indeed thought that the hardness of the DLP problem in $\mathbb{F}_{q^k}$ only depended on the bitlength of this field (namely $k \log_2(q)$) and not on k and q themselves. Using this assumption, it was quite simple to derive concrete bounds on ℓ, q and k to achieve some specific security level λ. For example, in the standard case $\lambda = 128$, a simple look at [39, Table 4.6] reveals that we must have $\log_2(\ell) \geqslant 256$ (and so $\log_2(q) \geqslant 256$ because of the Hasse's bound) and $k \log_2(q) \geqslant 3072$. In this case, parameters $q \sim \ell \sim 2^{256}$ and $k = 12$ are optimal and even more so because the choice of an even k that accepts several divisors leads to very efficient implementations of pairings and of the arithmetic in $\mathbb{F}_{q^k}$. This explains in large part the success of the so-called Barretto-Naehrig curves (BN) [6] (that perfectly match these parameters) which have become *de facto* the standard pairing curves in the literature.

Unfortunately, two recent papers [30,31] have shown that the assumption regarding $\mathbb{F}_{q^k}$ was wrong. We will discuss the details later but intu-

itively these results imply that the bitlength of the elements of $\mathbb{F}_{q^k}$ is no longer the good metric to estimate security in this group as it now depends on the shape of both integers q and k. We now face a somewhat chaotic situation where a 3000-bit finite field $\mathbb{F}_{q^k}$ may offer the same security level as a 5000-bit one provided that k and q have some specific properties. And $\mathbb{G}_T$ is not the only group concerned by these considerations as the parameter q has a direct impact on $\mathbb{G}_1$ and $\mathbb{G}_2$. Concretely, it is now sometimes necessary to significantly increase the parameter q (that becomes much larger than the advised minimal bound 2^{256}) to remain compatible with some values of k that enables efficient pairing computations. This is illustrated by the BLS12 curves promoted by Barbulescu and Duquesne's [4] for the standard 128-bits level of security. These curves lead to a parameter q of roughly 450 bits, *i.e.* 75% higher than the minimal bound for elliptic curves.

These examples are representative of the current strategy to select pairing parameters (see [4] for some alternative curves). The goal is indeed to find a nice compromise between the complexity of the different groups/operations. More specifically, it aims at providing parameters that would not significantly penalize one particular operation. This strategy thus makes the implicit assumption that cryptographic protocols present some kind of symmetry, between the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, but also between the entities that will perform the operations in these groups. This may be true in some specific scenarios but there are many others were this assumption is false.

As an example, let us consider the case of Enhanced Privacy ID (EPID) scheme introduced by Brickell and Li [12] and now massively used to secure Intel SGX enclave [2]. EPID is a variant of Direct Anonymous Attestation [10] with enhanced revocation capabilities, meaning that it is possible to revoke a secret key sk by simply adding a signature generated using sk to a revocation list. The next signatures will then have to contain a proof that they were issued using a different secret key. This is a nice feature but it implies to perform a high number of exponentiations in $\mathbb{G}_1$ [11], linear in the number n of revoked signatures, as illustrated in Table 1. Actually, this table shows a clear imbalance between the different groups as soon as the revocation list contains dozens of elements, a threshold that should be quickly reached in most scenarios. In those cases, we note that trying to stick to the minimal bound for q (thus decreasing the complexity of operations in $\mathbb{G}_1$), even if it significantly deteriorates the computational cost of a pairing, is a worthwhile goal as there is only one pairing to compute against roughly $6n$ exponentiations to generate

or to verify the signature. This is all the more true since this pairing is only computed by the verifier, an entity that is assumed, in the context of Direct Anonymous Attestation, to be much more powerful than the signer (usually a constrained device). To put it differently, we here need a curve that will optimize operations in $\mathbb{G}_1$ even if it is at the cost of a much more expensive pairing.

This scenario illustrates the limits of the global strategy for selecting parameters. The mainstream curves do not seem suitable here and we can hope for dramatic performance improvements by using a tailored curve. And this is not an isolated case as we explain in section 3.

Our contribution. The contribution of our paper is twofold. First, we investigate a different approach for selecting curve parameters. We indeed believe that standard families of curves, like BLS12, do not fit all cryptographic protocols and in particular impose a tradeoff (between the complexities of the different groups and operations) that is inappropriate in many contexts. Realizing that the era of optimality using Barreto-Naehrig curves is over, we choose to focus on the family of cryptographic protocols whose practicality depends on the implementation of the group $\mathbb{G}_1$. This family is quite large because cryptographers usually try to avoid as much as possible the other groups ($\mathbb{G}_2$ and $\mathbb{G}_T$) as the latter are much less efficient and even incompatible with some constrained devices (see *e.g.* [5]). We then look for curves optimizing the performance of $\mathbb{G}_1$, which leads us to the case of prime embedding degree k, a setting that has been overlooked for a long time despite being immune to Kim's and Barbulescu's attacks [30,31] mentioned above. We provide a security assessment, some benchmarks and a complexity evaluation of some curves from this setting that we compare to the most known alternatives. Our results show that the standard curves (with composite embedding degrees) increase by at least 65% the computational cost of an exponentiation in $\mathbb{G}_1$, compared to curves with prime k, while yielding elements in $\mathbb{G}_1$ that are 20% larger. Of course, those composite-embedding-degree curves come at the cost of a less expensive pairing but our analysis shows that this overhead is acceptable in the context we consider.

Based on these results we investigate new curves that would match our need. We find a new one that goes one step further in the quest for optimizing the performance of $\mathbb{G}_1$. We call this new curve, which constitutes our second contribution, `BW19-P286` as it was generated using the Brezing-Weng strategy [9], has embedding degree 19 and is defined over a 286-bit field $\mathbb{F}_q$. The use of such a small q, which is close to the

optimal bound 2^{256}, is particularly interesting for constrained devices (more specifically those with 32 -or less- bits of architecture) as it reduces the number of machine-words compared to the state-of-the-art.

In the end, our results show that Kim's and Barbulescu's attacks do not necessarily imply a large increase of the complexity of pairing-based protocols that would in particular rule out the latter for constrained devices. On the contrary, we prove that we can retain the original efficiency for some parties. Of course, this is done to the detriment of the other parties but we argue that there are few use-cases where all entities are equally powerful. We nevertheless do not claim that our curves fit all contexts and in particular we do believe that standard curves still remain relevant, in particular for protocols requiring to perform a large number of pairings.

Roadmap. In section 3 we describe some examples that justify our strategy for selecting curves. In section 4 we outline the strategy to assess the cost of the Discrete Logarithm Problem and the security of our curves and provide two tailored curves in section 5. Finally, we compared an implementation of the proposed curves with other curves in section 6.

2 Preliminaries

Let $q > 3$ be a prime number. The field having q elements is noted $\mathbb{F}_q$ and, for $n > 1$, the extension field having q^n elements is noted $\mathbb{F}_{q^n}$. When we compute discrete logarithms in $\mathbb{F}_{q^n}$, we mean solving the Discrete Logarithm Problem in the group $(\mathbb{F}_{q^n}\backslash\{0\}, \times)$.

2.1 Elliptic Curves

An elliptic curve E is the set of points (x, y) satisfying $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_q$ and $4a^3 + 27b^2 \neq 0$, enlarged with another point ∞, called point at infinity. This equation is called the Short Weierstrass Model of the curve. For $n \geqslant 1$, the set of point $(x, y) \in \mathbb{F}_{q^n}$ on E is noted $E(\mathbb{F}_{q^n})$. The set $E(\mathbb{F}_{q^n})$ can be equipped with a commutative internal law, with ∞ as identity element. We follow the convention of cryptographic literature and denote this group multiplicatively. For an integer ℓ coprime to q, we note $E(\mathbb{F}_{q^n})[\ell]$ the subgroup of $E(\mathbb{F}_{q^n})$ formed by points of order dividing ℓ, *i.e.* all points g such that $g^\ell = \infty$. The group $E(\mathbb{F}_{q^n})[\ell]$ is called the ℓ-torsion over $\mathbb{F}_{q^n}$. When we compute discrete logarithms in $E(\mathbb{F}_{q^n})[\ell]$, we mean solving the Discrete Logarithm Problem in the group $(E(\mathbb{F}_{q^n})[\ell], \cdot)$,

i.e. if h is a power of g, find x such that $h = g^x$. There is a minimal $k \geqslant 1$ such that $E(\mathbb{F}_{q^k})[\ell]$ is isomorphic to $(\mathbb{Z}/\ell\mathbb{Z})^2$, this integer is called the embedding degree of q (with respect to ℓ), it is the order of $q \pmod{\ell}$.

The Frobenius endomorphism is defined as $(x, y) \mapsto (x^q, y^q) \in \mathrm{End}(E)$. Its minimal polynomial is $X^2 - tX + q$, where t is aptly called the trace of the Frobenius, and the number of point on the curve is $q - t + 1$. The discriminant Δ of that polynomial is $\Delta = t^2 - 4q$. We can always write $\Delta = Df^2$, where $D < 0$ is square-free. D is called the Complex Multiplication discriminant. When the CM discriminant is small enough, there exists an endomorphism ϕ easily computable and acting as a multiplication on $E(\mathbb{F}_q)[\ell]$ when ℓ is a distinct prime form q, *i.e.* there exists an integer $\lambda > 0$ such that $\phi(g) = g^\lambda$ for all $g \in E(\mathbb{F}_q)[\ell]$.

The main advantage of using the endomorphism ϕ is to halve the computational cost of an exponentiation in $E(\mathbb{F}_q)[\ell]$ as evaluating ϕ is much more efficient than directly raising to the power λ. This is called the GLV method [22]. Suppose we want to compute g^a for a point $g \in E(\mathbb{F}_q)[\ell]$ and a random scalar $a \pmod{\ell}$. We proceed as follow: compute a_0 and a_1 such that $a = a_0 + a_1\lambda$ (*e.g.* the Euclidean division of a by λ) and compute $g^{a_0} \cdot \phi(g)^{a_1}$. The result is g^a:

$$g^{a_0} \cdot \phi(g)^{a_1} = g^{a_0} \cdot g^{a_1\lambda} = g^{a_0 + a_1\lambda} = g^a.$$

The size of a_0 and a_1 is expected to be half the size of ℓ as λ is a root of a degree 2 polynomial. The point $\phi(P)$ can be precomputed (if needed) and $g^{a_0} \cdot \phi(g)^{a_1}$ can be computed with any multi-exponentiation algorithm.

2.2 Bilinear Groups

Pairing-based cryptographic protocols consider a setting defined by three cyclic groups, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, of prime order ℓ (with respective identity element $1_{\mathbb{G}_1}$, $1_{\mathbb{G}_2}$ and $1_{\mathbb{G}_T}$), along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. for all $g \in \mathbb{G}_1, \tilde{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}/\ell\mathbb{Z}$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{a \cdot b}$;
2. for any $g \neq 1_{\mathbb{G}_1}$ and $\tilde{g} \neq 1_{\mathbb{G}_2}$, $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$;
3. the map e is efficiently computable.

As all these groups are of the same prime order, we know that there exist non-trivial homomorphisms $\varphi_1 : \mathbb{G}_1 \to \mathbb{G}_2$ and $\varphi_2 : \mathbb{G}_2 \to \mathbb{G}_1$. However, the latter may not be efficiently computable, which has a strong impact on cryptographic protocols and more specifically on the underlying computational assumptions. Following Galbraith, Paterson and Smart [21], this

has led cryptographers to distinguish types of pairings: type-1, where both φ_1 and φ_2 are efficiently computable, type-2, where only φ_2 is efficiently computable, and type-3, where no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction. All these types can be instantiated with elliptic curves but type-3 pairings are preferred in practice both for their efficiency and their ability to support some useful cryptographic assumptions, *e.g.* decisional Diffie-Hellman in groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

We also note that it is possible to consider bilinear groups of composite order. However, prime order bilinear groups are much more efficient [24] and can actually emulate most features of their composite-order counterparts [19].

Usually, when bilinear groups are instantiated over an elliptic curve, $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$, $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k})[\ell] \backslash \mathbb{G}_1$ and $\mathbb{G}_T \subset \mathbb{F}_{q^k}$.

3 Schemes with numerous computations in $\mathbb{G}_1$

Before providing details on the way we select elliptic curve parameters, we elaborate on the motivation of our work, namely the benefits of selecting such parameters based on the characteristics of the cryptographic protocols. We are more specifically interested in the family of cryptographic protocols whose complexity essentially depends on the efficiency of $\mathbb{G}_1$. This family may include protocols requiring to perform many exponentiations in $\mathbb{G}_1$, as is the case with the EPID scheme we discuss in the introduction, but also schemes where the most constrained entity only has to compute operations in $\mathbb{G}_1$, as in Direct Anonymous Attestation. These two primitives are today massively used in industrial products [40,2] and are thus meaningful examples of this family of cryptographic protocols. To illustrate that the latter is not restricted to authentication algorithms we will also consider the case of two cryptographic accumulators that would benefit from the tailored curves we propose in our paper.

Table 1 highlights the specific need of two anonymous authentication schemes [5,11] that are, to our knowledge, the most efficient of their kind. For [11], we use the proof of non-revocation described by the same authors in [12]. We note that most alternatives and variants (*e.g.* [35] for group signature) present similar features so our conclusions also apply to them. Table 1 shows that the size of the signature only depends on the one of $\mathbb{G}_1$ elements (and on ℓ) and that the signer only has to perform operations in $\mathbb{G}_1$. There are few pairings and operations in $\mathbb{G}_2$ to compute and only on the verifier side, which is usually considered as more powerful than the signer in those contexts. Cryptographic protocols with such kinds

of features are thus a good incentive for designing curves with efficient computations/elements in $\mathbb{G}_1$ even to the detriment of the complexity of the other groups.

Table 1. Complexity of some anonymous authentication schemes. $\mathbf{e}_i$ refers to an exponentiation in $\mathbb{G}_i$ and $\mathbf{P}$ to a pairing computation, n is the number of revoked signatures.

Primitive	Ref.	Signature elements	Operation Counts Sign	Verify
DAA	[5]	$5\mathbb{G}_1 + 2\mathbb{Z}/\ell\mathbb{Z}$	$6\mathbf{e}_1$	$4\mathbf{e}_1 + 3\mathbf{P}$
EPID	[11]	$3\mathbb{G}_1 + 6\mathbb{Z}/\ell\mathbb{Z}$ $+n(3\mathbb{G}_1 + \mathbb{Z}/\ell\mathbb{Z})$	$3\mathbf{e}_1 + 4\mathbf{e}_T$ $+6n\mathbf{e}_1$	$4\mathbf{e}_1 + 2\mathbf{e}_2 + 4\mathbf{e}_T$ $+1\mathbf{P} + 6n\mathbf{e}_1$

In Table 2, we consider two pairing-based accumulator schemes [13,20]. We recall that the point of an accumulator system is to project a large number of elements into a single short value, called the accumulator. Additionally, for each of these elements, it is possible to generate a short evidence, called witness, that the element has indeed been accumulated. In practice, there are essentially two kinds of entities, the one that needs to prove that an element has been accumulated (by computing the corresponding witness) and the one that checks this proof. We will then divide the public parameters of such systems between the ones (pk) necessary for the proof and the ones (vk) necessary for the verification.

Table 2. Complexity of some accumulators schemes. The latter are called set commitment schemes in [20]. Here $\mathbf{m}_1$ refers to a group operation in $\mathbb{G}_1$, n is a bound on the number of values to be accumulated and j is the number of values currently accumulated. The other notations are those from the previous table.

Ref.	Public Parameters pk	vk	Operation Counts Sign	Verify
[13]	$2n\mathbb{G}_1$	$n\mathbb{G}_2$	$j\mathbf{m}_1$	$2\mathbf{P}$
[20]	$n\mathbb{G}_1$	$n\mathbb{G}_2$	$(j-1)\mathbf{e}_1$	$1\mathbf{e}_2 + 2\mathbf{P}$

Here again, Table 2 shows a clear asymmetry between the prover and the verifier. The former is only impacted by the performance of $\mathbb{G}_1$ and so would clearly benefit from a curve tailored to optimize this group. This is all the more true that in the applications considered in [13] and [20], the prover is usually a user's device whereas the verifier is some

service provider that can reasonably be considered as more powerful. In particular, reducing the size of pk, even at the cost of a larger vk, seems to be a worthwhile goal.

4 Attacks solving the DLP

Most cryptographic schemes using bilinear groups rely on problems that are easier than the Discrete Logarithm Problem (DLP). Unfortunately, the concrete hardness of these problems is not known so the common approach to generate bilinear groups is to select parameters that yield three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ where the DLP is believed to be hard. The latter problem has indeed been extensively studied over the last 40 years and several algorithms were proposed to solve it.

The DLP on elliptic curves is called ECDLP (EC stands for Elliptic Curve) and is considered the hardest discrete logarithm problem to solve as only generic algorithms [38] are known and used, such as Baby-Step-Giant-Step [37] or Pollard-rho [36]. Moreover, the Pohlig-Hellman method [34] makes the DLP in a cyclic group of order n as hard as the DLP in a cyclic group of order p, for p the largest prime dividing n. Hence, for efficiency, the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ must be of prime order, *i.e.* ℓ is a prime number, and since the best variant of Pollard-rho [7] compute a discrete logarithm in at most $\sqrt{\pi\ell/4} \approx 0.886\sqrt{\ell}$ steps, $\log_2(\ell)$ must be at least 2λ for a security of λ bits.

A well-known value, called ρ-value, is used to describe the efficiency of the representation of elements of $\mathbb{G}_1$. It is computed as $\rho = \log(q)/\log(\ell)$ or as $\rho = \deg(q)/\deg(\ell)$ when q and ℓ are polynomial in $\mathbb{Q}[X]$. When $\rho = 1$, the curve is of prime order ℓ, this is the best case for the arithmetic efficiency on the curve (and so for $\mathbb{G}_1$).

While determining the size of $\mathbb{G}_1$ and $\mathbb{G}_2$ over the elliptic curve is pretty straightforward, doing the same for $\mathbb{G}_T$ over the finite field $\mathbb{F}_{q^k}$ is much harder! Indeed, discrete logarithms in $\mathbb{F}_{q^k}$ are computed in sub-exponential time (and sometimes even in quasi-polynomial time) by algorithms of the Number Field Sieve (NFS) familly. In the general case, it is difficult to evaluate the complexity of the NFS algorithm, and of its variants (see [27] for more details). To give an idea of the time-complexity of NFS, we need to introduce the L-notation, which is defined by:

$$L_{q^k}[\alpha, c] = \exp\left((c + o(1))(\ln q^k)^{\alpha}(\ln \ln q^k)^{1-\alpha}\right)$$

with $\alpha \in [0, 1]$ and $c > 0$. Intuitively, if $\alpha = 1$ then $L_{q^k}[\alpha, c]$ is exponential in $\log_2(q^k)$ whereas it is polynomial in $\log_2(q^k)$ if $\alpha = 0$. NFS-type algo-

rithms (of our concern) all have time-complexity $L_{q^k}[1/3, c]$ for c ranging from $\sqrt[3]{32/9}$ to $\sqrt[3]{96/9}$. The constant c plays an important role in the concrete (*i.e.* not asymptotic) world, as between 20 and 30 bits of security can be lost for the same size of $\mathbb{F}_{q^k}$ depending on the choices of q and k [4]. Two criteria determine which NFS-variant to use: whether q is special, *i.e.* if q is computed as the image of a polynomial of degree at least 3, and whether k is composite. Until recently, only curves with special q and composite k were consider as they offer the best performance for pairing computations. Barbulescu and Duquesne updated key size estimations in [4]. For a 128-bit security level, they urge to use a finite field of size at least $k \log_2(q) = 2930$ (respectively 3618, 5004) if NFS (respectively exTNFS, SexTNFS) is the best algorithm for computing discrete logarithms in $\mathbb{F}_{q^k}$ (the common practice was $k \log_2(q) = 3072$ [39]). To access curves' security, we will use the algorithm provided by Guillevic at `https://gitlab.inria.fr/tnfs-alpha/alpha`.

The only known methods for constructing curves with ρ close to 1 give the characteristic of the finite field as a polynomial (thus q is special), like Brezing-Weng constructions [9]; other general methods having ρ close to 2, like Cocks-Pinch constructions (see [26]). In this context, we will consider the recent bounds $[3072, 5376]$ on the finite field size provided by Guillevic [25] to achieve 128 bits of security.

If we have $\log_2(\ell) = 256$ to satisfy 128 bits of security on the curve side, we then know that $\log_2(q) = 256\rho$ and the finite field $\mathbb{F}_{q^k}$ has size $256\rho k$. Guillevic's bounds then give us:

$$3072 \leqslant 256\rho k \leqslant 5376,$$

which, together with the inequalities $1 \leqslant \rho \leqslant 2$, allows us to derive the set of potential values for $k : 6 \leqslant k \leqslant 21$. Concretely, this means that there is no point in considering values $k < 6$ as they are incompatible with the targeted 128-bit security level (for the range of ρ-values we consider) and selecting a value $k > 21$ would be an overkill.

So far, we have just managed to derive some bounds on the different parameters of the curves. Unfortunately, as we explain above, there is no simple choice within these bounds as security and efficiency of the resulting bilinear groups may significantly differ from one set of parameters to another. In particular, there is no linearity in the security evaluation as, for instance, the security of $\mathbb{F}_{q^k}$ is significantly higher in the "prime" cases $k \in \{11, 13\}$ than in the case $k = 12$. This means that we can select smaller q values (which improves performance of $\mathbb{G}_1$) in the former case. Unfortunately, a similar issue arises regarding efficiency of $\mathbb{F}_{q^k}$, but

with opposite conclusions, as non-prime k (especially even ones) yield more efficient pairings and group operations. It is thus necessary to make a choice between these different parameters, in particular in the case of cryptographic protocols with unbalanced complexities, such as the ones we consider in section 3. As the latter would benefit of fast $\mathbb{G}_1$ computations and short representations of its elements, we dedicate the next section to the selection of parameters that will optimize the performance of this group.

5 Curves optimizing operation in $\mathbb{G}_1$

The first group of the pairing $\mathbb{G}_1$ is defined as $E(\mathbb{F}_q)[\ell]$. We have an incentive to reduce the size of q as computations in $\mathbb{G}_1$ would be faster. To ensure security on the elliptic curve, we need $\log_2(q) \geqslant \log_2(\ell) \geqslant 256$. Thus q is at least a 5-machine-word on a 64-bit computer. Indeed, q has more than 257 bits, because a 256-bit field would imply $\rho = 1$ and the biggest known k for that ρ-value is 12, which does not ensure a 128-bit security in $\mathbb{G}_T$ [4].

We also want to be able to use the GLV method [22], that is, our curves should have a small Complex Multiplication discriminant. Usually GLV-curves are chosen either in the form $y^2 = x^3 + ax$ with a primitive fourth root of unity in $\mathbb{F}_q$ and CM discriminant -1, or in the form $y^2 = x^3 + b$ with a primitive third root of unity in $\mathbb{F}_q$ and CM discriminant -3.

As explained above, we cannot hope for better than 5 machine-words for q on 64-bit architecture. A 5-machine-word q means that $1 < \rho \leqslant 1.25$. Searching in the Taxonomy by Freeman, Scott and Teske [18], the curve we are looking for has embedding degree $k \in \{8, 11, 13, 16, 17, 19\}$.

The embedding degree 8 does not provide a secure finite field $\mathbb{F}_{q^k}$, as it is of at most $8 \times 1.25 \times 256 = 2560$ bits, and so is discarded. The embedding degree 16 corresponds to the well-known KSS family of pairing-friendly curves [29]. Barbulescu and Duquesne [4] state that q must be at least a 330-bit prime, *i.e.* a 6-machine-word prime, and they give such primes.

Thus we focus our search for curves from [18] having embedding degree $k \in \{11, 13, 17, 19\}$. All those curves correspond to Freeman, Scott and Teske Construction 6.6, which is a generalization of the Brezing and Weng construction. It defines the prime q, the Frobenius trace t and the order ℓ as polynomials in $\mathbb{Q}[X]$, where $\ell(X)$ is a cyclotomic polynomial dividing $q(X) - t(X) + 1$ and $q(X)^k - 1$.

5.1 Curves over a five-64-bit-machine-word prime field

Construction 6.6 from [18] is different given the residue $k \pmod 6$. Since we have $13 \equiv 19 \equiv 1 \pmod 6$ and $11 \equiv 17 \equiv 5 \pmod 6$, we only give the relevant cases of Construction 6.6.

In the case $k \equiv 1 \pmod 6$, the prime $q(X)$, the Frobenius trace $t(X)$ and the order $\ell(X)$ are given as:

$$q(X) = \frac{1}{3}(X+1)^2(X^{2k} - X^k + 1) - X^{2k+1},$$
$$t(X) = -X^{k+1} + X + 1 \quad \text{and} \quad \ell(X) = \Phi_{6k}(X),$$

and in the case $k \equiv 5 \pmod 6$, they are given as:

$$q(X) = \frac{1}{3}(X^2 - X + 1)(X^{2k} - X^k + 1) + X^{k+1},$$
$$t(X) = X^{k+1} + 1 \quad \text{and} \quad \ell(X) = \Phi_{6k}(X),$$

Plugging in the different values of k gives Table 3. Note that to find a curve, we need to find a $x_0 \in \mathbb{Z}$ such that $\ell(x_0)$ and $q(x_0)$ are prime integers. We choose x_0 satisfying $x_0 \equiv 2 \pmod 3$ so $q(x_0)$ is an integer. The last column of Table 3 is the search range of $\log_2(|x_0|)$ in $[256/\deg(\ell), 320/\deg(q)[$ so that $q(x_0)$ is a 5-machine-word integer and $\ell(x_0)$ is at least a 256-bit integer (for readability, the interval is given with rounded integer values).

Table 3. Parameters and search range for curves with $k \in \{11, 13, 17, 19\}$

k	$\deg(q)$	$\deg(\ell)$	ρ	$\log_2(\lvert x_0\rvert)$
11	24	20	1.20	$[12, 14[$
13	28	24	1.167	$[10, 12[$
17	36	32	1.125	$[8, 9[$
19	40	36	1.111	$[7, 8[$

After a computer search, we have found no solution for $k \in \{11, 17\}$. For $k = 13$, we found $x_0 = -2224$ and for $k = 19$, we found $x_0 = -145$. Inferring the naming convention used in relic-toolkit [3], the first curve is named `BW13-P310` and the second one `BW19-P286`. As Construction 6.6 curves have CM discriminant -3, both curves are of the form $y^2 = x^3 + b$.

Curve `BW13-P310` Setting $k = 13$ into Construction 6.6 [18] gives:

$$q(X) = \frac{1}{3}(X+1)^2(X^{26} - X^{13} + 1) - X^{27},$$
$$t(X) = -X^{14} + X + 1 \quad \text{and} \quad \ell(X) = \Phi_{78}(X).$$

Plugging in $x_0 = -2224$ yields a $q(x_0)$ of 310 bits and a $\ell(x_0)$ of 267 bits, thus $\rho = 1.161$. The corresponding $\mathbb{F}_{q^{13}}$ is a 4027-bit finite field.

To look for a b, we increase the value of $|b|$, check that the number of points on the curve $y^2 = x^3 + b$ is equal to $q(x_0) - t(x_0) + 1$. We found $b = -17$. So we defined the curve `BW13-P310` by the equation $y^2 = x^3 - 17$.

We also point out that the exact same curve has been given by Aurore Guillevic in [25]. She made a thorough security analysis and estimated that the cost of the DLP in the finite field $\mathbb{F}_{q^{13}}$ is 140 bits. Hence, the curve `BW13-P310` has a security of at least 128 bits.

Curve `BW19-P286` Setting $k = 19$ into Construction 6.6 [18] gives:

$$q(X) = \frac{1}{3}(X+1)^2(X^{38} - X^{19} + 1) - X^{39},$$
$$t(X) = -X^{20} + X + 1 \quad \text{and} \quad \ell(X) = \Phi_{114}(X).$$

Plugging in $x_0 = -145$ yields a $q(x_0)$ of 286 bits and a $\ell(x_0)$ of 259 bits, thus $\rho = 1.105$. The corresponding $\mathbb{F}_{q^{19}}$ is a 5427-bit finite field.

To look for a b, we do the same as before. The smallest $|b|$ is $b = 31$. So we defined the curve `BW19-P286` by the equation $y^2 = x^3 + 31$. To our knowledge, this curve has never been proposed in the literature.

To evaluate the cost of the DLP in $\mathbb{F}_{q^{19}}$, we follow the work of Guillevic [25], the same she did for the previous curve `BW13-P310`. To find the curve `BW19-P286` using Guillevic's Algorithm 3.1 [25], we plugged in the parameters $k = 19$, $D = 3$, $e_0 = 13$ and use the substitution $X \mapsto -X$.

Before running the estimating program on our parameters, we applied Variant 4 [25] to the polynomial $q(-X)$, yielding a polynomial $Q(X)$ such that $Q(u^3) = 3q(-u)$, for $u = -x_0 = 145$ and

$$Q(X) = (u+1)X^{13} + u^2X^{12} + X^7 + u(1-2u)X^6 + u^2 - 2u + 1.$$

Then we obtain that the cost of the DLP in $\mathbb{F}_{q^{19}}$ is 160 bits, thus providing `BW19-P286` with a security of at least 128 bits.

5.2 GLV endomorphism on BW13-P310 and BW19-P286

As stated in the preliminaries, the discriminant of the minimal polynomial of the Frobenius endomorphism can be written as $Df^2 = t^2 - 4q$, where $D < 0$ is the CM discriminant and t is the trace of the Frobenius. The endomorphism $\phi : (x, y) \mapsto (\omega x, y)$, with ω a primitive third root of unity, corresponds to an exponentiation $(\omega x, y) = (x, y)^\lambda$ in $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$ for q and ℓ distinct primes.

For both our curves, the CM discriminant is $D = -3$, thus we have $4q = t^2 + 3f^2$. Since $\omega \in \mathbb{F}_q$ is a primitive third root of unity, ω satisfies $\omega^2 + \omega + 1$. We can take $\omega = (\sqrt{-3} - 1)/2$, where $\sqrt{-3} \equiv t/f \pmod q$. Thus $\omega \equiv (t - f)/(2f) \pmod q$.

Similarly, since $\phi^3 = \mathrm{id}_E$ in $\mathrm{End}(E)$, we know that $\lambda \in \mathbb{Z}/\ell\mathbb{Z}$ satisfies the equation $\lambda^2 + \lambda + 1$, *i.e.* it can also be taken as $(\sqrt{-3} - 1)/2$. However, here, $\sqrt{-3} \equiv (t - 2)/f \pmod \ell$. Indeed, ℓ divides the number of points on the curve, so $q \equiv t - 1 \pmod \ell$ and $4(t - 1) \equiv t^2 + 3f^2 \pmod \ell$. Thus $\lambda \equiv (t - f - 2)/(2f) \pmod \ell$.

Note that in practice, adjustments may be needed as $(\omega x, y) = (x, y)^{\pm\lambda}$ or $(\omega^2 x, y) = (x, y)^{\pm\lambda}$.

In the case of BW13-P310, λ has bit-length 146, whereas it is only 137 in the case of BW19-P286.

Comments on BW13-P310 and BW19-286. We provide in the next section several benchmarks to compare our new curve with BW13-P310 but also with other curves from popular families. However, we can already note that BW13-P310 and BW19-P286 clearly match our strategy of optimizing the group $\mathbb{G}_1$ to the detriment of the other groups. In this respect, our new curve BW19-P286 goes one step further than BW13-P310 by reducing the size of q by roughly 25 bits. This difference is significantly amplified in the context of constrained devices as it results in less machine words. Indeed, even for 32-bit architecture, BW19-P286 yields a prime q with one less machine-word than BW13-P310, which clearly impacts performance, as illustrated below.

6 Implementation and comparison

We implemented the $\mathbb{G}_1$ arithmetic of both curves using relic-toolkit [3], and made some comparison with other curves already implemented in relic-toolkit and aiming the 128-bit security.

The curves selected from the framework are `BN-P446`, a Barreto-Naehrig curve [6] over a 446-bit prime field; `K16-P339`, a Kachisa-Schaefer-Scott curve of embedding degree 16 over a 339-bit field; `B12-P446`, a Barreto-Lynn-Scott curve of embedding degree 12 over a 446-bit field; and `CP8-P544`, a Cocks-Pinch curve [26] of embedding degree 8 over a 544 prime field. We chose the last curve as it is coming from recent works [26,25] that promote them for the 128-bit security level. We included a BN, BLS and KSS curves in our table as those families of curves are well-known and were updated by Barbulescu and Duquesne [4]. Also note that, setting aside our curves, only the curve `K16-P339` was implemented by us in the framework.

6.1 Operation in $\mathbb{G}_1$

In Table 4 we compare the cost of one exponentiation in the group $\mathbb{G}_1$ by compiling the relic-toolkit either for x64 architecture or for x86 architecture with a word size of 32 bits. Times are given in microseconds (the benchmarking number of iterations was 10^6) and computations were done on a laptop equipped with a Intel Core i7-6600u CPU at 2.60 GHz.

Table 4. Benchmark for one exponentiation in $\mathbb{G}_1$.

	Curve	`BW19-P286`	`BW13-P310`	`K16-P339`	`B12-P446`	`BN-P446`	`CP8-P544`
prime q bit size		286	310(+8%)	339(+19%)	446(+56%)	446(+56%)	544(+90%)
64-bit	words	5	5	6	7	7	9
	time (μs)	293	304(+4%)	482(+65%)	611(+109%)	855(+192%)	1 058(+261%)
32-bit	words	9	10	11	14	14	17
	time (μs)	1 010	1 220(+21%)	1 664(+65%)	2 510(+149%)	3 600(+256%)	4 180(+314%)

This table shows a clear relation, almost quadratic, between complexity and the number of words necessary to represent q. It also highlights the downside of Barreto-Naehrig curves that generate elliptic curves of prime order $\ell \sim q$. Indeed, what was considered as an advantage (prime order curves make group membership tests trivial) turns out to be a strong limitation as it forces ℓ to grow unnecessarily. This negatively impacts both exponentiation (as the exponents are roughly 75% greater than those of the other curves) and the size of scalars.

In all cases, this table shows that our curve `BW19-P286` offers the best performance for $\mathbb{G}_1$, in particular for architecture smaller than 64 bits. It at least halves the complexity of exponentiations in $\mathbb{G}_1$ compared to

mainstream curves such as B12-P446 and also significantly decreases the size of group elements, which clearly fits the needs of some cryptographic protocols such as the ones we presented in section 3.

6.2 Operation in $\mathbb{G}_2$

The operations in $\mathbb{G}_2$ are unfortunately the ones that are the most impacted by our choice of prime embedding degree. Indeed, BW13-P310 and BW19-P286 have $\mathbb{G}_2$ defined over $\mathbb{F}_{q^{13}}$ and $\mathbb{F}_{q^{19}}$ respectively, whereas B12-P446, BN-P446 and CP8-P544 all have $\mathbb{G}_2$ defined over $\mathbb{F}_{q^2}$ thanks to quartic or sextic twists, and K16-P339 has $\mathbb{G}_2$ defined over $\mathbb{F}_{q^4}$ thanks to a quartic twist. As all curves are usually expressed with the same model using the same system of coordinates, only the cost of the multiplication in the extension impacts the cost of the operations in $\mathbb{G}_2$. Using a Karatsuba-like implementation, the multiplication in $\mathbb{F}_{q^k}$ is roughly $k^{\log_3 2}$ times as expensive as the one in $\mathbb{F}_q$.

6.3 Pairing computation

The computation of the pairing is usually split between two parts: the evaluation of the Miller loop and the final exponentiation. Here we give computation for a multiple of the Optimal Ate Pairing [41], since we picked the final exponentiation from Kim, Kim and Cheon [32]. The values in Table 5 are from [25,26], completed with the ones from below.

Let $\mathbf{m}_k$, $\mathbf{s}_k$, $\mathbf{i}_k$, $\mathbf{f}_k$ respectively denote a multiplication, a square, an inversion, a Frobenius map (*i.e.* the q-th power map) over $\mathbb{F}_{q^k}$. We drop the index when the operation is over $\mathbb{F}_q$ (*i.e.* $\mathbf{m} = \mathbf{m}_1$). As $k \in \{13, 19\}$ is prime, we estimate $\mathbf{m}_k = \mathbf{s}_k = k^{\log_2 3}\mathbf{m}$ with a Karatsuba-like implementation and $\mathbf{f}_k = (k-1)\mathbf{m}$ as in [26].

Miller loop Using Equation (7) from [25], Guillevic gives a lower bound on the cost of the Miller loop. For both BW13-P310 and BW19-P286, the optimal ate Miller loop has length u^2+up+p^2, as it is a multiple of ℓ [41].

For BW13-P310, the length of the Miller loop is $u^2 + up + p^2$, where $u = 2224$ is a 12-bit integer with Hamming weight 4 and p is a 310-bit prime. From her Equation (7) [25], Guillevic obtains $949\mathbf{m} + 313\mathbf{m}_{13} + 177\mathbf{s}_{13} + 5\mathbf{f}_{13} + 2\mathbf{i}_{13}$. Substituting $\mathbf{m}_{13} = \mathbf{s}_{13} = 59\mathbf{m}$ and $\mathbf{f}_{13} = 12\mathbf{m}$ in that formula yields a lower bound on the cost of the optimal ate Miller loop, *i.e.* $29919\mathbf{m} + 2\mathbf{i}_{13}$.

For BW19-P286, the length of the Miller loop is $u^2 + up + p^2$, where $u = 145$ is a 8-bit integer with Hamming weight 3 and p is a 286-bit prime.

From the same equation as Guillevic [25], we obtain $912\mathbf{m} + 212\mathbf{m}_{19} + 115\mathbf{s}_{19} + 5\mathbf{f}_{19} + 2\mathbf{i}_{19}$. Substituting $\mathbf{m}_{19} = \mathbf{s}_{19} = 107\mathbf{m}$ and $\mathbf{f}_{19} = 18\mathbf{m}$ in that formula yields a lower bound on the cost of the optimal ate Miller loop, *i.e.* $35991\mathbf{m} + 2\mathbf{i}_{19}$.

Final exponentiation As usual, the final exponentiation $(q^k - 1)/\ell$ of the Optimal Ate Pairing is split between an easy part $(q^k - 1)/\Phi_k(q)$ and a hard part $\Phi_k(q)/\ell$. Since $k \in \{13, 19\}$ is prime, the easy part is simply $q - 1$, costing $\mathbf{f}_k + \mathbf{i}_k$. For the hard part, Kim, Kim and Cheon [32] noticed that $\Phi_k(q)/\ell$ can be decompose in base q to make use of the Frobenius and the coefficients can be reduced by looking for a short vector in a specifically designed lattice. However, instead of raising to the power $\Phi_k(q)/\ell$, this method [32] raises to a multiple power $m\Phi_k(q)/\ell$.

More precisely, they write

$$m\frac{\Phi_k(q)}{\ell} = \sum_{i=0}^{k-2} a_i q^i$$

and find the $k - 1$ coefficients $(a_i)_{0 \leqslant i \leqslant k-2}$ as the shortest vector in the dim-$(k - 1)$ lattice spanned by the lines of the following matrix:

$$\begin{bmatrix} \frac{\Phi_k(q)}{\ell} & 0 & 0 & \cdots & 0 \\ -q & 1 & 0 & \cdots & 0 \\ -q^2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \\ -q^{k-2} & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Then, they compute the Frobenius of the element they want to exponent, up to the $(k - 1)$-th q-power, costing $(k - 2)\mathbf{f}_k$.

If the exponents a_i's were longer, we would have needed $(2^{k-1} - k)\mathbf{m}_k$ to compute all combinations of $(k - 1)$ Frobenius powers. However, we do not use all of these combinations, only roughly $\mathcal{O}(\log_2 q)$ of them. Finally the length of the multi-exponent is $\max_i\{\lfloor \log_2 a_i \rfloor\}$, resulting in an average final exponentiation costing

$$(k-1)\mathbf{f}_k + (\mathcal{O}(\log_2 q) + \max_i\{\lfloor \log_2 a_i \rfloor\})\mathbf{m}_k + \max_i\{\lfloor \log_2 a_i \rfloor\}\mathbf{s}_k + \mathbf{i}_k,$$

omitting some inversion due to the sign of some a_i's.

For `BW13-P310`, the value of $\max_i\{\lfloor \log_2 a_i \rfloor\}$ is 287 and 8 of the 12 a_i's are negative. Only 191 different combinations of Frobenius powers are used and it costs $341\mathbf{m}_{13}$ to compute them. Also, there are 5 positions

(in the binary expansion) where all the a_i's have their bit set to 0, resulting in no multiplication at those positions for the multi-exponentiation, that thus requires $282\mathbf{m}_{13} + 287\mathbf{s}_{13}$. Combining everything yields an final exponentiation cost of $12\mathbf{f}_{13} + 623\mathbf{m}_{13} + 287\mathbf{s}_{13} + 9\mathbf{i}_{13}$, *i.e.* $53\ 834\mathbf{m} + 9\mathbf{i}_{13}$.

For `BW19-P286`, the value of $\max_i\{\lfloor \log_2 a_i \rfloor\}$ is 271 and 12 of the 18 a_i's are negative. Only 222 different combinations of Frobenius powers are used and it costs $1028\mathbf{m}_{19}$ to compute them. The multi-exponentiation requires $271(\mathbf{m}_{19} + \mathbf{s}_{19})$. Combining everything yields an final exponentiation cost of $18\mathbf{f}_{19} + 1299\mathbf{m}_{19} + 271\mathbf{s}_{19} + 13\mathbf{i}_{19}$, *i.e.* $160\ 824\mathbf{m} + 13\mathbf{i}_{19}$.

Table 5. Operation count for Miller loop and Final exponentiation

Curve	`BW19-P286`	`BW13-P310`	`K16-P339`	`B12-P446`	`BN-P446`	`CP8-P544`
Miller loop	$35\ 991\mathbf{m} + 2\mathbf{i}_{19}$	$29\ 919\mathbf{m} + 2\mathbf{i}_{13}$	$7\ 691\mathbf{m}$	$7\ 805\mathbf{m}$	$11\ 620\mathbf{m}$	$4\ 502\mathbf{m}$
Final exp.	$160\ 824\mathbf{m} + 13\mathbf{i}_{19}$	$53\ 834\mathbf{m} + 9\mathbf{i}_{13}$	$18\ 235\mathbf{m}$	$7\ 723\mathbf{m}$	$5\ 349\mathbf{m}$	$7\ 056\mathbf{m}$
Total	$196\ 815\mathbf{m} + 15\mathbf{i}_{19}$	$83\ 753\mathbf{m} + 11\mathbf{i}_{13}$	$25\ 926\mathbf{m}$	$15\ 528\mathbf{m}$	$16\ 969\mathbf{m}$	$11\ 558\mathbf{m}$

From Table 5, the cost of the pairing for `BW19-P286` is roughly 12 times higher than the one for `B12-P446`. However, doing the benchmark on both finite field gives a multiplication twice faster on the 286-bit finite field (90 ns) than the 446-bit one (190 ns). Hence, we estimate that the pairing over `BW19-P286` is 6 times slower than the pairing over `B12-P446`.

Conclusion

In this paper, we have given an incentive to change the way pairing-friendly elliptic curve are constructed by shifting the optimization away from the balance between all operations (group exponentiation and pairing) towards only some operations (that might be used by constrained entities involved in cryptographic protocols).

Thus, we focused on elliptic curves with a fast exponentiation in the first pairing group upon noticing that the instantiation of some cryptographic protocols, *e.g.* Group Signature-like schemes, would benefit from such curves.

Along the way, we have described a new curve that is particularly relevant for cryptographic protocols extensively using exponentiation in the first pairing group. That curve is twice faster in that group and its pairing computation is reasonably six times slower compared to a BLS curve over a 446-bit field.

We leave to future work the investigation of other protocols-curves dependencies.

Acknowledgements

The authors are grateful for the support of the ANR through project ANR-19-CE39-0011-04 PRESTO and project ANR-18-CE-39-0019-02 MobiS5.

References

1. Masayuki Abe, Fumitaka Hoshino, and Miyako Ohkubo. Design in type-i, run in type-iii: Fast and scalable bilinear-type conversion using integer programming. In *CRYPTO 2016*, LNCS. Springer, 2016.
2. Guy AlLee. EPID for IoT Identity. https://img.en25.com/Web/McAfeeE10BuildProduction/{a6dd7393-63f8-4c08-b3aa-89923182a7e5}_EPID_Overview_Public_2016-02-08.pdf, 2016.
3. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.
4. Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 2019.
5. Amira Barki, Nicolas Desmoulins, Saïd Gharout, and Jacques Traoré. Anonymous attestations made practical. In *ACM WiSec 2017*, pages 87–98. ACM, 2017.
6. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*. Springer Berlin Heidelberg, 2005.
7. Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. On the correct use of the negation map in the pollard rho method. In *PKC 2011*, LNCS. Springer, 2011.
8. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *SCN 2010*, LNCS. Springer, 2010.
9. Friederike Brezing and Annegret Weng. Elliptic curves suitable for pairing based cryptography. *Des. Codes Cryptogr.*, 2005.
10. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM CCS 2004*, pages 132–145. ACM, 2004.
11. Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. In *IEEE SocialCom*. IEEE Computer Society, 2010.
12. Ernie Brickell and Jiangtao Li. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *IEEE Trans. Dependable Secur. Comput.*, 9(3):345–360, 2012.
13. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC 2009*, LNCS, 2009.
14. Sanjit Chatterjee and Alfred Menezes. Type 2 structure-preserving signature schemes revisited. In *ASIACRYPT 2015*, LNCS. Springer, 2015.
15. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the CLT13 multilinear map. *J. Cryptology*, 2019.

16. Jung Hee Cheon, Changmin Lee, and Hansol Ryu. Cryptographic multilinear maps and their cryptanalysis. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 2018.
17. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO 2013*, LNCS. Springer, 2013.
18. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 2010.
19. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.
20. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *J. Cryptology*, 32(2):498–546, 2019.
21. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 2008.
22. Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *CRYPTO 2001*, LNCS. Springer, 2001.
23. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*, LNCS. Springer, 2013.
24. Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In *ACNS 2013*, LNCS. Springer, 2013.
25. Aurore Guillevic. A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In *PKC 2020*, LNCS. Springer, 2020.
26. Aurore Guillevic, Simon Masson, and Emmanuel Thomé. Cocks-pinch curves of embedding degrees five to eight and optimal ate pairing computation. *Des. Codes Cryptogr.*, 88(6), 2020.
27. Aurore Guillevic and François Morain. Discrete Logarithms. In *Guide to pairing-based cryptography*. CRC Press - Taylor and Francis Group, 2016.
28. Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptology*, 2004.
29. Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In *Pairing 2008*, LNCS. Springer, 2008.
30. Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *CRYPTO 2016*, LNCS. Springer, 2016.
31. Taechan Kim and Jinhyuck Jeong. Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In *PKC 2017*, LNCS. Springer, 2017.
32. Taechan Kim, Sungwook Kim, and Jung Hee Cheon. On the final exponentiation in tate pairing computations. *IEEE Trans. Inf. Theory*, 59(6):4033–4041, 2013.
33. Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *ACM STOC*, 1991.
34. Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 1978.
35. David Pointcheval and Olivier Sanders. Short randomizable signatures. In *CT-RSA 2016*, LNCS. Springer, 2016.
36. John M. Pollard. Monte carlo methods for index computation (mod p). 1978.

37. Daniel Shanks. Class number, a theory of factorization, and genera. In *1969 Number Theory Institute*, pages 415–440. American Mathematical Society, 1971.
38. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT '97*, LNCS. Springer, 1997.
39. Nigel P Smart. Algorithms, key size and protocols report, ECRYPT - CSA, `http://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf`, 2018.
40. TCG. `https://trustedcomputinggroup.org/authentication/`, 2015.
41. Frederik Vercauteren. Optimal pairings. *IEEE Trans. Inf. Theory*, 2010.