Compact hardware for computing the Tate pairing over 128-bit-security supersingular curves

Nicolas Estibals

Compact hardware for computing the Tate pairing over 128-bit-security supersingular curves

Nicolas Estibals

Équipe-projet CARAMEL, LORIA, Nancy Université / CNRS / INRIA
Campus Scientifique, BP 239 54506 Vandoeuvre-lès-Nancy Cedex, France

Abstract. This paper presents a novel method for designing compact yet efficient hardware implementations of the Tate pairing over supersingular curves in small characteristic. Since such curves are usually restricted to lower levels of security because of their bounded embedding degree, aiming for the recommended security of 128 bits implies considering them over very large finite fields. We however manage to mitigate this effect by considering curves over field extensions of moderately-composite degree, hence taking advantage of a much easier tower field arithmetic. This technique of course lowers the security on the curves, which are then vulnerable to Weil descent attacks, but a careful analysis allows us to maintain their security above the 128-bit threshold.
As a proof of concept of the proposed method, we detail an FPGA accelerator for computing the Tate pairing on a supersingular curve over $\mathbb{F}_{3^{5\cdot 97}}$, which satisfies the 128-bit security target. On a mid-range Xilinx Virtex-4 FPGA, this accelerator computes the pairing in 2.2 ms while requiring no more than 4755 slices.

Keywords: Tate pairing, supersingular elliptic curves, FPGA implementation.

1 Introduction

Pairings were first introduced in cryptography in 1993 by Menezes, Okamoto, & Vanstone [36] and Frey & Rück [24] as an attack against the elliptic curve discrete logarithm problem (ECDLP) for some families of curves over finite fields. Since then, constructive properties of pairings have also been discovered and exploited in several cryptographic protocols: starting independently in 2000 with Joux's one-round tripartite Diffie–Hellman key agreement [31] and Sakai–Ohgishi–Kasahara cryptosystem [46], many others have followed, such as Mitsunari–Sakai–Kasahara broadcast encryption scheme [39], Boneh–Franklin identity-based encryption [12] or Boneh–Lynn–Shacham short signature [13] for instance. Pairings nowadays being the cornerstone of various protocols, their efficient implementation on a wide range of targets became a great challenge, especially on low-resource environments.

Although many FPGA implementations of pairing accelerators have been proposed [2, 6, 7, 9, 30, 34, 43, 47], none of them allows to reach the AES-128 security level. However, recent ASIC implementations of pairings over Barreto–Naehrig (BN) [4] curves with 128 bits of security have been published [22, 33]. The main difficulty for computing a pairing at the 128-bit security level is to implement an efficient arithmetic over a quite large finite field.

In contrast with the ASIC implementation, we chose to implement pairings over supersingular elliptic curves over small-characteristic finite fields so as to benefit from the many optimizations available in the literature. As a drawback, since supersingular curves are restricted to low embedding degrees, this implies considering unbalanced settings, where the curve offers potentially much more security than the required 128 bits. Nonetheless we took advantage of this excess of security and defined our curves over finite fields of composite extension degree: on the one hand, the curves might be weaker because of, for instance, the Gaudry–Hess–Smart attack [17,26,27]; on the other hand, the arithmetic algorithm can really benefit from this tower field structure. This article is devoted to the demonstration that this compromise is very effective in the context of a low-resources hardware implementation.

After a reminder on the Tate pairing and its security in a general context (Section 2), we present the consequences on security of defining an elliptic curve over a composite-extension field (Section 3). We then detail the algorithms for computing the Tate pairing over such curves in Section 4 and present a low-area FPGA accelerator implementing these algorithms for a test-case curve in Section 5. Finally we report our performance results and compare them against other implementations from the literature (Section 6) and conclude in Section 7.

2 Definition and security of the Tate pairing

Given an elliptic curve E defined over a finite field $\mathbb{F}_q$, take ℓ a prime number dividing the cardinal of the curve $\#E(\mathbb{F}_q)$. The embedding degree k of E is then defined as the smallest integer such that $\ell \mid q^k - 1$, that is to say such that the group of ℓ-th roots of unity $\mu_\ell = \{x \in \overline{\mathbb{F}_q} \mid x^\ell = 1\}$ is in $\mathbb{F}_{q^k}^*$. Assuming further that $k > 1$ and that there are no points of order ℓ^2 in $E(\mathbb{F}_{q^k})$, we can then define the Tate pairing over E as the map:

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mathbb{F}_{q^k}^* / \left(\mathbb{F}_{q^k}^*\right)^\ell \cong \mu_\ell,$$

where $E(\mathbb{F}_q)[\ell] = \{P \in E(\mathbb{F}_q) \mid [\ell]P = \mathcal{O}\}$ denotes the $\mathbb{F}_q$-rational ℓ-torsion subgroup. The embedding degree k, also called security multiplier in this context, acts as a cursor to adjust the size of the multiplicative group $\mathbb{F}_{q^k}^*$ with respect to that of $\mathbb{F}_q$, which directly constrains $\#E(\mathbb{F}_q)$ to Hasse's bounds, therefore limiting the achievable values of ℓ. Given that the discrete logarithm problem (DLP) is exponential in the subgroup $E(\mathbb{F}_q)[\ell]$ but subexponential in the finite field $\mathbb{F}_{q^k}^* \supset \mu_\ell$ (*cf.* Section 2.2), one might want to choose a curve giving a security

multiplier k that balances the security on both the input and the output of the Tate pairing.

As we are targeting the AES-128 security level, elliptic curves with an embedding degree between 12 and 15 seem to be a good choice. Barreto–Naehrig (BN) curves are a family of such curves with prime cardinal $\ell = \#E(\mathbb{F}_q)$ and embedding degree $k = 12$ [4]; as a result BN curves perfectly balance the security between the ℓ-torsion and μ_ℓ at the 128-bit level. However, since BN curves are defined over prime fields, computing a pairing over them requires expensive modular arithmetic, which is far less better-suited to hardware implementation than arithmetic over small-characteristic finite fields. Last but not least, BN curves are ordinary curves: point doubling and tripling formulae are not as efficient as in the supersingular case in characteristic 2 and 3 respectively.

As a consequence, we chose to consider supersingular elliptic curves even if their embedding degree is bounded by 6 [3]. Due to this bound, the security on the curve will be too high with respect to the security on μ_ℓ. We however decided to take advantage of this: using finite fields with composite extension degree will decrease the security on the curves but make the field arithmetic better suited to low-resource hardware implementations. Those points will be detailed and quantified in the next two sections.

We now detail the definition, security and computation of the Tate pairing over the considered supersingular elliptic curves.

2.1 Pairing over supersingular elliptic curves

Our study focuses on pairings on supersingular curves over finite fields $\mathbb{F}_q$ with $q = p^m$ and $p = 2$ or 3. We thus define the two following families [3]:

$$\begin{aligned} E_{2,b}/\mathbb{F}_2 &: y^2 + y = x^3 + x + b, \text{ where } b \in \{0, 1\}; \text{ and} \\ E_{3,b}/\mathbb{F}_3 &: \quad y^2 = x^3 - x + b, \quad \text{where } b = \pm 1. \end{aligned}$$

When m is coprime to 2 and 6 in characteristic 2 and 3 respectively, the cardinal of those curves reaches the Hasse bounds:

$$\begin{aligned} \#E_{2,b}(\mathbb{F}_q) &= 2^m \pm 2^{\frac{m+1}{2}} + 1, \\ \#E_{3,b}(\mathbb{F}_q) &= 3^m \pm 3^{\frac{m+1}{2}} + 1. \end{aligned}$$

Moreover, their embedding degree is 4 and 6 in characteristic 2 and 3, respectively. Thanks to their supersingularity, there exists a distortion map over those elliptic curves, mapping the $\mathbb{F}_q$-rational ℓ-torsion group to another subgroup of $E(\mathbb{F}_{q^k})[\ell]$:

$$\delta : E(\mathbb{F}_q)[\ell] \to E(\mathbb{F}_{q^k})[\ell],$$

which is used to define the modified Tate pairing as:

$$\hat{e} : \begin{cases} E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_q)[\ell] & \to \mathbb{F}_{q^k}^* / \left(\mathbb{F}_{q^k}^*\right)^\ell \cong \mu_\ell \\ (P, Q) & \mapsto e(P, \delta(Q)) \end{cases}.$$

One can furthermore show that $\hat{e}$ is not degenerate. We refer the reader to [3] and [9, Table I] for the mathematical details of pairing construction over supersingular curves.

2.2 Attacks against pairings over supersingular curves

The security of the pairing is determined by the difficulty of the discrete logarithm problem (DLP) on the input curve and on the output multiplicative group.

Since ℓ is a prime, the best known algorithm to attack the DLP on the ℓ-torsion is Pollard's ρ method [42], which requires an average of $\sqrt{\pi\ell/2}$ group operations. As Duursma *et al.* showed in [19, 25], we should take into account the group of automorphisms on the curve, which has order 24 and 12 in characteristic 2 and 3, respectively, [48, Chap. III, §10] as well as the m iterated Frobenius endomorphisms $(x, y) \mapsto (x^{p^i}, y^{p^i})$, for $0 \leqslant i < m$ as they allow to speed up Pollard's ρ method by a $\sqrt{m \cdot \#\mathrm{Aut}(E)}$ factor. All in all the average cost of Pollard's method on

$E(\mathbb{F}_q)[\ell]$ is:

$$\begin{cases} \sqrt{\frac{\pi\cdot\ell}{48m}} & \text{if } p = 2, \text{ and} \\ \sqrt{\frac{\pi\cdot\ell}{24m}} & \text{if } p = 3. \end{cases}$$

Additionally, one may attack the DLP on $\mu_\ell \subset \mathbb{F}_{q^k}^*$; this is the fundamental idea behind the attacks of Menezes, Okamoto, & Vanstone [36] and Frey & Rück [24]. Since the ℓ-th roots of unity are defined in the multiplicative group of a finite field, the DLP may be attacked by sieving algorithms. In our case, where the characteristic p is 2 or 3, one can use the function field sieve (FFS) [1]; the complexity of this attack is subexponential:

$$\exp\left(\left(\frac{32}{9} + o(1)\right)^{\frac{1}{3}} \cdot \log^{\frac{1}{3}} q^k \cdot \log\log^{\frac{2}{3}} q^k\right).$$

If we consider our 128-bit security level target, we need to take m between 1100 and 1200 in characteristic 2 and around 500 in characteristic 3.

3 Elliptic curves over composite-extension fields

We examine, in this section, the consequences on security of defining supersingular elliptic curves over a finite field of the form $\mathbb{F}_{q^n}$, where $q = p^m$, n is a small integer and m a prime. This corresponds to substituting q^n for q and $m \cdot n$ for m in the previous section.

It is important to remark that such elliptic curves defined over composite-extension fields have already been described for cryptographic use under the name Trace-Zero Variety (TZV) [23]. Applying the Weil descent to $E(\mathbb{F}_{q^n})$, we obtain an isomorphic variety $W_E(\mathbb{F}_q)$ which is also isomorphic to the product

$E(\mathbb{F}_q) \times B(\mathbb{F}_q)$ where $B(\mathbb{F}_q)$ is the TZV. It is a variety defined over the base field $\mathbb{F}_q$ which might also be represented as the quotient $E(\mathbb{F}_{q^n})/E(\mathbb{F}_q)$. As we consider in this work an ℓ-torsion subgroup of $E(\mathbb{F}_{q^n})$ which is not contained in $E(\mathbb{F}_q)$, this ℓ-torsion is a subgroup of the corresponding TZV. In the context of pairings, TZVs have also been studied, chiefly for point compression [16, 44, 45].

3.1 The Gaudry–Hess–Smart attack

As soon as one defines a curve on a field of composite extension degree, one should also consider other attacks: the Weil descent can indeed be applied on those curves and have some "destructive facets." The Weil descent allows one to map an elliptic curve defined over $\mathbb{F}_{q^n}$ to the Jacobian of a curve of genus at least n over $\mathbb{F}_q$.

Thus the discrete logarithm problem on the elliptic curve defined over $\mathbb{F}_{q^n}$ might be transported to the DLP on the Jacobian of a genus-n curve over $\mathbb{F}_q$. This last DLP can then be solved using an index calculus algorithm. Gaudry, Hess, & Smart have shown that this attack (GHS) runs in $\tilde{O}(q^{2-\frac{2}{n}})$ in some cases (Weil restrictions) [27]. More generally Gaudry [26] and Diem [17] showed that this also holds in the general case, but with a very bad dependency in n (hidden in the big-O notation).

3.2 The static Diffie–Hellman problem

Recent studies [28,32] showed that defining a curve over a finite field of composite extension degree makes it weaker regarding the static Diffie–Hellman problem (SDH). The SDH problem on a curve consists in: given two points $P, [d]P \in E(\mathbb{F}_q)$ (where d is a secret integer) and an oracle $Q \mapsto [d]Q$, compute $[d]R$ where R is randomly chosen point.

The cryptographic consequence of solving SDH problem is breaking the Diffie–Hellman key exchange protocol when one participant never changes his private key, as it occurs in the El Gamal encryption scheme for instance [20].

Granger discovered the best known algorithm that solves the SDH problem on elliptic curves defined over a field of composite extension degree $\mathbb{F}_{q^n}$ with $O(q^{1-\frac{1}{n+1}})$ calls to the oracle and in $\tilde{O}(q^{1-\frac{1}{n+1}})$ time [28].

One should notice that the attacker not only needs a great computational power but also a great number of calls to the oracle: a simple but efficient protection against this attack is revoking a key after a certain amount of use.

3.3 Finding curves with 128-bit security level

To the best of our knowledge, the literature does not mention any other attack on curves over fields of composite extension degree.

In order to find suitable curves for our method, we enumerated all the supersingular curves of characteristic 2 and 3 on fields with moderately-composite extension degrees $m \cdot n$ ($n < 15$) large enough for the 128-bit security level. We

then evaluated an approximation (constants hidden in big-O are not taken into account) of the computation time of each of the attacks mentioned in the paper: Pollard's ρ, FFS, GHS and SDH. A selection of curves reaching the 128-bit level of security is given in Table 1; since that is not necessarily a security issue for all protocols, we also present curves that are not resistant to Granger's SDH attack.

				Cost of the attacks (bits)			
q	n	b	$\log_2 \ell$	Pollard's ρ	FFS	GHS	SDH
2^{1117}	1	1	1076	531	128	–	–
2^{367}	3	1	698	342	128	489	275
2^{227}	5	1	733	359	129	363	189
2^{163}	7	1	753	370	129	279	142
2^{127}	9	1	487	236	130	225	114
2^{103}	11	1	922	454	129	187	94
2^{89}	13	0	1044	515	164	130	82
2^{73}	15	0	492	239	136	127	68
3^{503}	1	1	697	342	132	–	–
$\mathbf{3^{97}}$	**5**	**−1**	**338**	**163**	**130**	**245**	**128**
3^{67}	7	−1	612	300	129	182	92
3^{53}	11	−1	672	330	140	152	77
3^{43}	13	1	764	376	138	125	63

Table 1. Different curves and their security in bits against the different known attacks. A security of N bits means that approximately 2^N operations are required to perform the attack.

The main difficulty in computing Table 1 is to factor the cardinal of the different curves because they contains more than 350 digits in characteristic 2 and 240 in characteristic 3. Luckily those cardinals are the Aurifeuillean factors of Cunningham numbers and many of them are referenced in the factor tables maintained by Wagstaff [49] and Leyland [35].

The security estimations given in Table 1 confirm the intuition: the more composite the extension degree of the field of definition, the more effective the attacks using Weil descent, until they become the best attack on the curves.

As a proof of concept, we finally chose to implement the pairing over the supersingular curve $E_{3,-1}$ over $\mathbb{F}_{3^{5\cdot 97}}$, as this curve has an embedding degree equal to 6 and is resistant to all the attacks, even for the SDH problem.

4 Computation of the Tate pairing over composite-extension fields

As we have identified some curves that allow us to reach the 128-bit level of security, we now focus on the algorithms for computing the pairing over such curves.

4.1 Algorithms for computing the Tate pairing

The computation of the Tate pairing is split into two parts: Miller's loop [37,38] and a final exponentiation in the multiplicative group $\mathbb{F}^*_{q^{k\cdot n}}$.

Many improvements of Miller's algorithm have been published since its discovery. Duursma & Lee adapted it to exploit the simple point-tripling formulae in characteristic 3 by turning the double-and-add into a triple-and-add algorithm [18]. Furthermore Barreto *et al.* put forward the η_T approach which divides by two the length of the loop by exploiting the action of the Verschiebung on the ℓ-torsion [5].[1] Those improvements and a careful implementation of the arithmetic of the extension over $\mathbb{F}_{q^{k\cdot n}}$ leads to the algorithms presented by Beuchat *et al.* in [6,8].

To implement the pairing of our test case, we chose the unrolled loop algorithm in [8, Algorithm 5] because it minimizes the number of multiplications on the field of definition $\mathbb{F}_{q^n}$ which represents the major cost on a field large enough to reach the AES-128 security level. Moreover this algorithm requires only additions, multiplications and cubings over $\mathbb{F}_{q^n}$ but not any cube rooting; therefore it represents a substantial saving in hardware resources requirements.

We have now determined the sequence of operations in $\mathbb{F}_{q^n}$ to compute the η_T pairing over $\mathbb{F}_{q^n}$. Nonetheless we want to design compact hardware to execute them: the datapath of a circuit directly handling elements of $\mathbb{F}_{q^n}$ would be very large. Therefore we take advantage of the composite extension degree of our field of definition and implement the pairing as sequence of operations over $\mathbb{F}_q$: the datapath of a coprocessor dealing with elements of $\mathbb{F}_q$ only will be much smaller. Thus we have to express the arithmetic of $\mathbb{F}_{q^n}$ in terms of operation over $\mathbb{F}_q$ in an efficient way.

4.2 Representation and computation over the extension

Pairing computation requires a large number of multiplications. Using normal basis would thus be very harmful. As a consequence $\mathbb{F}_{q^n}$ is represented using a polynomial basis: $\mathbb{F}_{q^n} \cong \mathbb{F}_q[X]/(f(X))$ where f is a degree-n irreducible polynomial over $\mathbb{F}_q$. Hence an element of $\mathbb{F}_{q^n}$ is represented as a polynomial of degree at most $n-1$ over $\mathbb{F}_q$, and operations over $\mathbb{F}_{q^n}$ are mapped to operations over $\mathbb{F}_q[X]$ followed if necessary by a reduction modulo f.

[1] The η_T pairing is in fact a power of the actual Tate pairing but the conversion between the two is free [6].

The irreducible polynomial f could be taken among all irreducible polynomials of degree n over $\mathbb{F}_q$ but we restricted this choice to polynomials over $\mathbb{F}_p$ in order to avoid multiplications over $\mathbb{F}_q$ during the different reductions modulo f. This is possible because n is coprime to m. We also chose f to have a low Hamming weight, *i.e.* a trinomial or a pentanomial, so as to further reduce the cost of the reductions.

Frobenius automorphism over $\mathbb{F}_{q^n}$. During the pairing computation, many iterated applications of the Frobenius, *i.e.* p^i-th powering, are required. By linearity of this operation, we have:

$$(a_0 + a_1 X + \cdots + a_{n-1} X^{n-1})^{p^i} = a_0^{p^i} + a_1^{p^i} X^{p^i} + \cdots + a_{n-1} X^{(n-1)\cdot p^i}.$$

Moreover we have that $X^{p^n} \equiv 1 \pmod{f}$ because f is defined over $\mathbb{F}_p$. Therefore computing the i-th iterated Frobenius over $\mathbb{F}_{q^n}$ is tantamount to computing the i-th iterated Frobenius over all coefficients and then applying a linear combination on them that only depends on the value of $i \bmod n$.

Multiplications over $\mathbb{F}_{q^n}$. Multiplication is the most expensive operation and it can be greatly optimized by using subquadratic multiplication schemes. Choosing the best algorithm to compute the products of two degree-$(n-1)$ polynomials depends on many criteria and we studied how different solutions fit our case.

Many subquadratic multiplication algorithms can be used: Karatsuba, Montgomery's Karatsuba-like formulae [21,40], or CRT-based algorithms [14,15]. The common point between those algorithms is that they can all be expressed as the linear combination of a set of products of linear combinations of the coefficients of the operands.

The Toom–Cook algorithm and its variants cannot be used easily in the case of polynomials over low-characteristic fields, as it is based on an evaluate–interpolate scheme. To be efficient, evaluation points, their inverse, and their successive powers should have a small representation. However, we cannot find enough "simple elements" in low-characteristic fields: taking interpolation points in $\mathbb{F}_q$ instead of $\mathbb{F}_p$ will increase the number of multiplications and defeat the whole point of the method.

Furthermore, as we will see in Section 5.1, additions do not have a negligible cost when compared to multiplications as it is often assumed in estimations of multiplication complexity. Thus we have to express the formulae given by the different algorithms and count the total number of operations of each type.

Inversion over $\mathbb{F}_{q^n}$. During the final exponentiation step of the pairing computation, an inversion over $\mathbb{F}_{q^n}$ has to be carried out. Because there is only one inversion in the whole pairing computation, there is no gain to dedicate specific hardware resources to speed up its computation. However, thanks to the Itoh–Tsujii algorithm [29] which consists in applying Fermat's little theorem,

the inversion over $\mathbb{F}_{q^n}$ is computed with $(n-1)\cdot m$ applications of the Frobenius in $\mathbb{F}_{q^n}$, some multiplications over $\mathbb{F}_{q^n}$ and one inversion over $\mathbb{F}_q$. We also used another Itoh–Tsujii's algorithm to compute this last inversion over $\mathbb{F}_q$ and then do not need any other inversion since inversion over $\mathbb{F}_p$ is the identity when $p = 2$ or 3.

4.3 Our test case: $\mathbb{F}_{3^{5\cdot 97}}$

We chose to construct the extension for our test case as $\mathbb{F}_{3^{5\cdot 97}} \cong \mathbb{F}_{3^{97}}[X]/(X^5 - X + 1)$, $\mathbb{F}_{3^{97}}$ itself being represented as $\mathbb{F}_3[t]/(t^{97} + t^{16} - 1)$. Thus we evaluated multiplication over the extension cost thanks to different algorithms (*cf.* Table 2):

- the quadratic and so-called schoolbook method;
- one-level Karatsuba, where the sub-products are computed using the schoolbook method;
- recursive Karatsuba, where the sub-products are also computed thanks to Karatsuba algorithm;
- Montgomery's Karatsuba-like formulae [40];
- algorithm based on the Chinese Remainder Theorem (CRT) by Cenk & Özbudak [14] (*cf.* Section A for detailed algorithm).

Since $n = 5$ is odd, Montgomery's trick [40, Section 2.3] for applying the Karatsuba formulae can be used and saves one extra sub-product.

As we have now expressed a variety of algorithms for multiplication over $\mathbb{F}_{3^{5\cdot 97}}$, choosing one of them is a matter of algorithm–architecture co-design. Indeed, timing for each algorithm heavily depends on:

- the cost of multiplication on $\mathbb{F}_{3^{97}}$ compared to the addition,
- the data dependencies, and
- the scheduling of the operations in regards to the memory architecture.

Algorithm	Multiplications over $\mathbb{F}_{3^{97}}$	Additions over $\mathbb{F}_{3^{97}}$	Add./Mul. Ratio
Schoolbook	25	24	0.96
One-level Karatsuba (Montgomery's trick)	21	29	1.38
Recursive Karatsuba	15	39	2.60
Recursive Karatsuba (Montgomery's trick)	14	43	3.07
Montgomery's Karatsuba-like formulae [40]	13	54	4.153
Cenk & Özbudak [14]	**12**	**53**	**4.42**

Table 2. Cost of different multiplication algorithms over $\mathbb{F}_{3^{5\cdot 97}}$

Finally, it turned out that the algorithm by Cenk & Özbudak [14] best fitted our arithmetic coprocessor (*cf.* Section 5). In conclusion, the overall cost of the arithmetic over the extension field $\mathbb{F}_{3^{5\cdot 97}}$ is presented in Table 3. Table 4 summarizes the number of operations over the field $\mathbb{F}_{3^{97}}$ and its extension $\mathbb{F}_{3^{5\cdot 97}}$ needed to perform Miller's loop and the final exponentiation from [8].

		$\times$	$+$	$(.)^3$
Addition		–	5	–
Multiplication		12	53	–
Iterated Frobenius, $(.)^{3^i}$	where $i \equiv 0 \pmod 5$	–	–	$5i$
	where $i \equiv 1 \pmod 5$	–	5	$5i$
	where $i \equiv 2 \pmod 5$	–	6	$5i$
	where $i \equiv 3 \pmod 5$	–	8	$5i$
	where $i \equiv 4 \pmod 5$	–	7	$5i$
Inverse		41	129	484

Table 3. Cost of the arithmetic over $\mathbb{F}_{3^{5\cdot 97}}$ in terms of operations over $\mathbb{F}_{3^{97}}$

	$\times$	$+$	$(.)^3$	$1/.$
$\mathbb{F}_{3^{5\cdot 97}}$	3104	13127	4123	1
$\mathbb{F}_{3^{97}}$	37289	253314	21099	–

Table 4. Count of operations for full-pairing computation over $\mathbb{F}_{3^{5\cdot 97}}$, and the corresponding cost over $\mathbb{F}_{3^{97}}$

5 Hardware accelerator for computing the Tate pairing

5.1 An arithmetic coprocessor over $\mathbb{F}_q$

As we have now reduced the pairing computation to a sequence of operations over $\mathbb{F}_q$ with $q = p^m$, we need a coprocessor able to perform additions, multiplications and Frobenius (squarings and cubings) over this field. To this intent, we chose the coprocessor that Beuchat *et al.* developed for the final exponentiation in [10].

The architecture of this coprocessor is reproduced in Fig. 1 and is composed of three units running in parallel: a register file implemented by means of a dual-ported RAM, a unit performing additions and Frobenius applications, and a parallel-serial multiplier. Several direct feedback paths exist between the inputs and outputs of the units, for instance allowing a product to be used in an

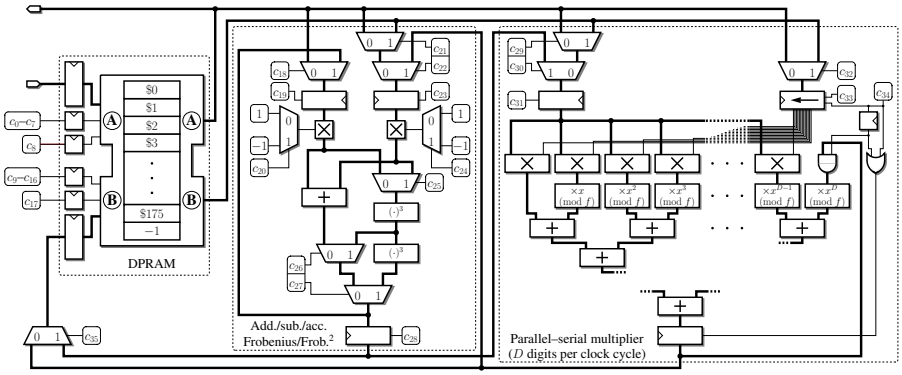

Fig. 1. Finite field coprocessor over $\mathbb{F}_q$

addition without having to go through the register file: this allows us to save time while decreasing the pressure on the memory, which is a major bottleneck of the architecture.

Frobenius computations are quite scarce in the overall pairing algorithm (*cf.* Table 4) but long sequences of iterated squarings or cubings occur several times. The coprocessor is designed to fit this observation: the addition unit shares most of its datapath with a Frobenius unit which can carry out both single and double applications of the Frobenius in one clock cycle. One should also notice that there is a direct feedback loop from its output to one of its inputs so as to further speed up sequences of Frobenius.

Products are processed in a parallel-serial fashion: at each cycle the first operand is multiplied by D coefficients of the second operand. The complete multiplication over $\mathbb{F}_{p^m}$ is then computed in $\left\lceil \frac{m}{D} \right\rceil$ clock cycles. D is a parameter of the processor and is chosen as trade-off between computation time of the multiplication and the operating frequency (a large value of D lengthens the critical path and this deteriorates the frequency).

In our case of computing the Tate pairing over $\mathbb{F}_{3^{5 \cdot 97}}$, we chose $D = 14$. The product on $\mathbb{F}_{3^{97}}$ then takes 7 clock cycles, *i.e.* 7 times longer than an addition. Given this cost ratio between multiplications and additions, the multiplication algorithm over $\mathbb{F}_{3^{5 \cdot 97}}$ by Cenk & Özbudak fit best the coprocessor, that is to say we managed to find a scheduling of the algorithm that hides all the additions behind the 12 multiplications over $\mathbb{F}_{3^{97}}$. A multiplication algorithm with less sub-products and more additions would not yield a better execution time since the bottleneck would be in the memory access. Indeed memory ports are near to be saturated in our scheduling of Cenk & Özbudak's algorithm.

5.2 Micro- and macrocode

Considering the total number of multiplications over $\mathbb{F}_q$ (*cf.* Table 4) and their cost, the pairing needs a minimum of 260 000 clock cycles to be calculated.

During those cycles, the 36 control bits (the c_i's in Fig. 1) should be set: this represents a total amount of 10 Mbit of memory for the pairing program. Thus we cannot store those control bits directly in an instruction memory: it would use up much more resources than the coprocessor itself.

In order to reduce instruction memory requirements, we implemented two levels of code. In the lower one, the microcode, we implemented the arithmetic over the extension $\mathbb{F}_{3^{5\cdot 97}}$. These operations are called in a macro-program that computes the actual pairing. Given that the non-reduced pairing is computed thanks to Miller's loop, we also constructed a loop mechanism on the macrocode.

Finally the implementation of the Tate pairing over $E(\mathbb{F}_{3^{5\cdot 97}})$ is a sequence of 464 macro-operations which takes 428 853 clock cycles to be executed. Although microcoding implies a loss of parallelism, it allows us to drastically reduce the size of the instruction memory, which now fits in 24 kbit.

The register file is split into two parts: the first one contains 32 macro-variables (elements of $\mathbb{F}_{3^{5\cdot 97}}$) and the second serves as a scratch space of 16 temporary variables (elements of $\mathbb{F}_{3^{97}}$) for use inside the microcode. Macro-variables are blocks of 5 consecutive addresses in the register file that are accessed in the microcode thanks to a windowed address mechanism. Since each element of $\mathbb{F}_3$ is represented by 2 bits, the total amount of RAM used is 33 kbit.

6 Results and comparisons

We prototyped and synthesized our design on Xilinx mid-range Virtex 4 and also on Spartan-3's, which are more suited to embedded systems. Place-and-route results show that the coprocessor uses 4755 slices and seven 18 kbit RAM blocks of a Virtex-4 (xc4vlx25-11) clocked at 192 MHz, finally computing our test-case pairing in no more than 2.11 ms. Performance for the low-end FPGA are more modest but still interesting: on a Spartan 3 (xc3s1000-5) running at 104 MHz, this pairing can be computed in 4.1 ms using 4713 slices.

To the best of our knowledge, this design is the first FPGA implementation of a pairing reaching 128 bits of security; thus we compared our design to FPGA implementations of less secure pairings (Table 5), along with ASIC (Table 6) and software (Table 7) implementations of 128-bit security pairings.

The literature about pairing computation on FPGAs only focuses on low-security pairings because they already reach the limit of the available FPGA resources. Indeed the designs presented in [2, 6, 7, 9, 47] have a datapath that handles the field of definition of their respective curves and thus increasing the security means increasing the designs' area. In contrast our approach allows us to "split" elements of the field of definition into smaller parts and thus achieve a smaller area: the coprocessor is very compact compared to the other published architectures. However we have to pay the price of security in terms of computation time: computing a pairing over $E(\mathbb{F}_{3^{5\cdot 97}})$ (128 bits of security) with our processor is 130 times slower than computing one over $E(\mathbb{F}_{3^{313}})$ (109 bits) with Beuchat *et al.*'s hardware [9]. It is however 20 times smaller.

	Curve	Sec. (bits)	FPGA	Area (slices)	Freq. (MHz)	Time (μs)	Area×time (slices.s)
Barenghi *et al.* [2]	$E(\mathbb{F}_{p_{512}})$	87	xc2v8000-5	33857	135	1610	54.5
Shu *et al.* [47]	$E(\mathbb{F}_{2^{457}})$	88	xc4vlx200-10	58956	100	100.8	5.94
Beuchat *et al.* [9]	$E(\mathbb{F}_{2^{457}})$	88	xc4vlx100-11	44223	215	7.52	0.33
Beuchat *et al.* [6]	$E(\mathbb{F}_{2^{459}})$	89	xc2vp20-6	8153	115	327	2.66
Beuchat *et al.* [6]	$E(\mathbb{F}_{3^{193}})$	89	xc2vp20-6	8266	90	298	2.46
Beuchat *et al.* [9]	$E(\mathbb{F}_{3^{193}})$	89	xc4vlx200-11	47260	179	9.33	0.44
Shu *et al.* [47]	$E(\mathbb{F}_{2^{557}})$	96	xc4vlx200-10	37931	66	675.5	25.62
Beuchat *et al.* [9]	$E(\mathbb{F}_{2^{557}})$	96	xc4vlx200-11	55156	139	13.2	0.73
Beuchat *et al.* [9]	$E(\mathbb{F}_{3^{239}})$	97	xc4vlx200-11	66631	179	11.5	0.77
Beuchat *et al.* [9]	$E(\mathbb{F}_{2^{613}})$	100	xc4vlx200-11	62418	143	15.1	0.95
Beuchat *et al.* [9]	$E(\mathbb{F}_{2^{691}})$	105	xc4vlx200-11	78874	130	18.8	1.48
Beuchat *et al.* [9]	$E(\mathbb{F}_{3^{313}})$	109	xc4vlx200-11	97105	159	16.9	1.64
This paper	$\boldsymbol{E(\mathbb{F}_{3^{5\cdot 97}})}$	**128**	**xc4vlx25-11** **xc3s1000-5**	**4755** **4713**	**192** **104**	**2227** **4113**	**10.59** **19.38**

Table 5. Tate pairing computation on FPGA.

	Platform	Area without RAM	RAM	Frequency	Time
Fan *et al.* [22]	130 nm ASIC	113 kGates	32 kbits	204 MHz	2.91 ms
Kammler *et al.* [33]	130 nm ASIC	97 kGates	64 kbits	308 MHz	15.8 ms

Table 6. Performance of some ASIC accelerators for pairings over BN-curves of AES-128 security level

The first ASIC implementations of pairings with 128 bits of security were presented in [22, 33]. The two implementations use BN-curves so as to exploit their optimal embedding degree $k = 12$ while targeting 128 bits of security. Although we did not synthesize our design on ASIC, a very rough and pessimistic estimation places our coprocessor around the 100-kGate mark, not counting the register file. That is to say roughly the same area as required by the two accelerators presented in Table 6. We also use 33 kbit of dual-ported RAM: a bit more than Fan *et al.* and half of the amount used by Kammler *et al.* As a result, our architecture seems to be very comparable with the ones from [22, 33] in terms of area, and its performance is also very closed to the ASICs' one.

	Curve	Processor	Time (ms)
Beuchat *et al.* [11]	Supersingular over $\mathbb{F}_{2^{1223}}$	2.4 GHz Intel Core2	11.9
	Supersingular over $\mathbb{F}_{3^{509}}$	2.4 GHz Intel Core2	7.59
Naehrig *et al.* [41]	257 bit BN curve	2.4 GHz Intel Core2	1.87

Table 7. Computation of pairing at the AES-128 security level in software

Finally we compared our results against single-core software implementations of 128-bits pairings over supersingular curves [11] and BN curves [41]. Even though, we targeted our implementation to embedded systems and low-resource hardware, our timings are very comparable to that of the software implementations: specific hardware for small-characteristic finite field arithmetic proves to be very efficient when compared to software implementations.

7 Conclusion

We presented a compact hardware implementation of a pairing reaching 128 bits of security, which is perfectly suited for embedded systems. To this end, we showed that the Tate pairing on supersingular curves over composite-extension field is a pertinent solution, even though their embedding degree k could be deemed too small at first glance. This also demonstrates that the efficiency of the underlying arithmetic plays a key role in pairing computation, and should be taken into account, right along with the size of the base field and the embedding degree, when designing pairing-based cryptosystems.

Furthermore, the idea to use curves defined over finite fields $\mathbb{F}_{q^n}$ of moderately-composed extension degree might be exploited in other areas of cryptography. While targeting the AES-128 level of security, the attacks based on Weil descent do not introduce extra weaknesses on the curve as long as n is kept small enough. This is an interesting result in itself: expanding the fauna of pairing-friendly curves suited to the 128-bit security level is indeed very relevant for cryptography. Moreover, computations on such curves can be carried out in a more efficient and parallel way, which yields better overall performances.

An interesting development of this work is to implement this idea in characteristic 2. Indeed, arithmetic over binary fields is simpler than in characteristic 3; as a consequence, characteristic 2 might also be a good choice, even though the embedding degree is even lower. We are planning to explore this direction in the near future.

Implementing the pairing on all the supersingular elliptic curves shown in Table 1 would also give a better coverage of the area–time trade-off for computing pairings with 128 bits of security: the more composite the extension degree, the smaller the base field $\mathbb{F}_q$ and thus the coprocessor. Additionally, in our approach, products over $\mathbb{F}_q$ are performed thanks to a quadratic scheme but the algorithms used for multiplications over $\mathbb{F}_{q^n}$ are subquadratic; therefore using a larger n for a same size of the field $\mathbb{F}_{q^n}$ might lead to a more efficient multiplication.

Furthermore, Cesena has noticed that the extra structure in curves defined over a composite-degree extension field—or TZVs—leads to a natural parallelization of Miller's algorithm [16]. It might be of interest to design a more parallel accelerator exploiting this fact. Such a circuit might achieve a lower latency for computing the Tate pairing with 128 bits of security at the cost of a larger silicon footprint.

Last but not least, the method presented in this article might scale to higher levels of security. For instance, the curve $E_{3,1}(\mathbb{F}_{3^{17\cdot 67}})$ reaches 192 bits of security,

while keeping the hardware requirements to a minimum. Finding other such curves and comparing them against higher-embedding-degree ordinary curves might help finding the crossover point between the two and assessing the actual relevance of supersingular elliptic curves in the context of low-resource pairing-based cryptography.

Acknowledgements

We sincerely thank Jérémie Detrey and Pierrick Gaudry for their insightful discussions and accurate comments that helped us to improve this work.

We would like to thank the anonymous reviewer for pointing out the following references [16, 23, 44, 45].

The author also wants to thank Paul Leyland and Sam W. Wagstaff for maintaining the factor tables of Cunningham numbers.

References

1. Adleman, L.M.: The function field sieve. In: Algorithmic NNumber Theory Symposium – ANTS I. pp. 108–121. No. 877 in Lecture Notes in Computer Science, Springer (1994)
2. Barenghi, A., Bertoni, G., Breveglieri, L., Pelosi, G.: A FPGA coprocessor for the cryptographic Tate pairing over $\mathbb{F}_p$. In: Proceedings of the Fourth International Conference on Information Technology: New Generations (ITNG'08). IEEE Computer Society (2008)
3. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. pp. 354–369. No. 2442 in Lecture Notes in Computer Science, Springer (2002)
4. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S.E. (eds.) Selected Areas in Cryptography – SAC. Lecture Notes in Computer Science, vol. 3897, pp. 319–331. Springer (2005)
5. Barreto, P., Galbraith, S., Ó hÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007)
6. Beuchat, J.L., Brisebarre, N., Detrey, J., Okamoto, E., Rodríguez-Henríquez, F.: A comparison between hardware accelerators for the modified Tate pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$. In: Galbraith, S., Paterson, K. (eds.) Pairing-Based Cryptography – Pairing 2008. pp. 297–315. No. 5209 in Lecture Notes in Computer Science, Springer (2008)
7. Beuchat, J.L., Doi, H., Fujita, K., Inomata, A., Ith, P., Kanaoka, A., Katouno, M., Mambo, M., Okamoto, E., Okamoto, T., Shiga, T., Shirase, M., Soga, R., Takagi, T., Vithanage, A., Yamamoto, H.: FPGA and ASIC implementations of the η_T pairing in characteristic three. Computers and Electrical Engineering 36(1), 73–87 (Jan 2010)
8. Beuchat, J.L., Brisebarre, N., Detrey, J., Okamoto, E., Shirase, M., Takagi, T.: Algorithms and arithmetic operators for computing the η_t pairing in characteristic three. IEEE Transactions on Computers – Special Section on Special-Purpose Hardware for Cryptography and Cryptanalysis 57(11), 1454–1468 (Nov 2008)

9. Beuchat, J.L., Detrey, J., Estibals, N., Okamoto, E., Rodríguez-Henríquez, F.: Fast architectures for the η_T pairing over small-characteristic supersingular elliptic curves (2009), cryptology ePrint Archive, Report 2009/398
10. Beuchat, J.L., Detrey, J., Estibals, N., Okamoto, E., Rodríguez-Henríquez, F.: Hardware accelerator for the Tate pairing in characteristic three based on Karatsuba-Ofman multipliers. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2009. pp. 225–239. No. 5747 in Lecture Notes in Computer Science, Springer (2009)
11. Beuchat, J.L., López-Trejo, E., Martínez-Ramos, L., Mitsunari, S., Rodríguez-Henríquez, F.: Multi-core implementation of the Tate pairing over supersingular elliptic curves. In: Garay, J., Miyaji, A., Otsuka, A. (eds.) Cryptology and Network Security – CANS 2009. pp. 413–432. No. 5888 in Lecture Notes in Computer Science, Springer (2009)
12. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. pp. 213–229. No. 2139 in Lecture Notes in Computer Science, Springer (2001)
13. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. Journal of Cryptology 17(4), 297–319 (2004)
14. Cenk, M., Özbudak, F.: Efficient multiplication in $\mathbb{F}_{3^{\ell m}}, m \geqslant 1$ and $5 \leqslant \ell \leqslant 18$. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. pp. 406–414. No. 6055 in Lecture Notes in Computer Science, Springer (2008)
15. Cenk, M., Özbudak, F.: On multiplication in finite fields. Journal of Complexity 26(2), 172–186 (2010)
16. Cesena, E.: Pairing with supersingular Trace Zero Varieties revisited (2008), cryptology ePrint Archive, Report 2008/404
17. Diem, C.: On the discrete logarithm problem in class groups of curves. Mathemathics of computation (to appear)
18. Duursma, I., Lee, H.S.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Laih, C. (ed.) Advances in Cryptology – ASIACRYPT 2003. pp. 111–123. No. 2894 in Lecture Notes in Computer Science, Springer (2003)
19. Duursma, I.M., Gaudry, P., Morain, F.: Speeding up the discrete log computation on curves with automorphisms. In: Lam, K.Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 1716, pp. 103–121. Springer (1999)
20. El-Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology – CRYPTO 1985. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984)
21. Fan, H., Hasan, M.A.: Comments on Montgomery's "Five, Six, and Seven-Term Karatsuba-Like Formulae". IEEE Transactions on Computers 56(5), 716–717 (May 2007)
22. Fan, J., Vercauteren, F., Verbauwhede, I.: Faster $\mathbb{F}_p$-arithmetic for Cryptographic Pairings on Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2009. pp. 240–253. No. 5747 in Lecture Notes in Computer Science, Springer (2009)
23. Frey, G.: Applications of arithmetical geometry to cryptographic constructions. In: Proceedings of the Fifth International Conference on Finite Fields and Applications (Augsburg, 1999). pp. 128–161. Springer (2001)
24. Frey, G., Rück, H.G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Mathematics of Computation 62(206), 865–874 (1994)

25. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Improving the parallelized Pollard lambda search on anomalous binary curves. Math. Comput. 69(232), 1699–1705 (2000)
26. Gaudry, P.: Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. Journal of Symbolic Compution 44(12), 1690–1702 (2009)
27. Gaudry, P., Hess, F., Smart, N.: Constructive and destructive facets of Weil descent on elliptic curves. Journal of Cryptology 15(1), 19–46 (2002)
28. Granger, R.: On the static Diffie–Hellman problem on elliptic curves over extension fields. In: ASIACRYPT. Lecture Notes in Computer Science, Springer (2010), to appear.
29. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases. Information and Computation 78(3), 171–177 (1988)
30. Jiang, J.: Bilinear pairing (Eta_T Pairing) IP core. Tech. rep., City University of Hong Kong – Department of Computer Science (May 2007)
31. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) Algorithmic Number Theory – ANTS IV. pp. 385–394. No. 1838 in Lecture Notes in Computer Science, Springer (2000)
32. Joux, A., Vitse, V.: Elliptic curve discrete logarithm problem over small degree extension fields. Application to the static Diffie–Hellman problem on $E(\mathbb{F}_{q^5})$ (2010), cryptology ePrint Archive, Report 2010/157
33. Kammler, D., Zhang, D., Schwabe, P., Scharwaechter, H., Langenberg, M., Auras, D., Ascheid, G., Mathar, R.: Designing an ASIP for cryptographic pairings over Barreto-Naehrig curves. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2009. pp. 254–271. No. 5747 in Lecture Notes in Computer Science, Springer
34. Kerins, T., Marnane, W., Popovici, E., Barreto, P.: Efficient hardware for the Tate pairing calculation in characteristic three. In: Rao, J., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. pp. 412–426. No. 3659 in Lecture Notes in Computer Science, Springer (2005)
35. Leyland, P.: Cunningham numbers, http://www.leyland.vispa.com/numth/factorization/cunningham/main.htm
36. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
37. Miller, V.S.: Short programs for functions on curves (1986), IBM, Thomas J. Watson Research Center
38. Miller, V.S.: The Weil pairing, and its efficient calculation. J. Cryptology 17(4), 235–261 (2004)
39. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundamentals E85–A(2), 481–484 (Feb 2002)
40. Montgomery, P.L.: Five, six, and seven-term Karatsuba-like formulae. IEEE Transactions on Computers 54(3), 362–369 (March 2005)
41. Naehrig, M., Niederhagen, R., Schwabe, P.: New software speed records for cryptographic pairings (2010), cryptology ePrint Archive, Report 2010/186
42. Pollard, J.: Monte Carlo methods for index computation (mod p). Mathematics of computation pp. 918–924 (1978)
43. Ronan, R., Ó hÉigeartaigh, C., Murphy, C., Scott, M., Kerins, T.: Hardware acceleration of the Tate pairing on a genus 2 hyperelliptic curve. Journal of Systems Architecture 53, 85–98 (2007)

44. Rubin, K., Silverberg, A.: Supersingular abelian varieties in cryptology. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 336–353. Springer (2002)
45. Rubin, K., Silverberg, A.: Using abelian varieties to improve pairing-based cryptography. Journal of Cryptology 22(3), 330–364 (2009)
46. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: 2000 Symposium on Cryptography and Information Security (SCIS2000), Okinawa, Japan. pp. 26–28 (Jan 2000)
47. Shu, C., Kwon, S., Gaj, K.: Reconfigurable computing approach for Tate pairing cryptosystems over binary fields. IEEE Transactions on Computers 58(9), 1221–1237 (Sep 2009)
48. Silverman, J.H.: The Arithmetic of Elliptic Curves. No. 106 in Graduate Texts in Mathematics, Springer Verlag (1986)
49. Wagstaff, S.: The Cunningham Project, http://homes.cerias.purdue.edu/~ssw/cun/index.html

A Multiplication algorithm over $\mathbb{F}_{3^{5m}}$

Given a characteristic-3 finite field $\mathbb{F}_{3^m}$ with $5 \nmid m$, its degree-5 extension $\mathbb{F}_{3^{5m}}$, and $\alpha \in \mathbb{F}_{3^{5m}}$ such that $\alpha^5 = \alpha - 1$, we present explicit formulae for multiplication over $\mathbb{F}_{3^{5m}}$ in Algorithm 1. These formulae come from the CRT-based algorithm presented by Cenk & Özbudak in [14].

Algorithm 1 Multiplication in $\mathbb{F}_{3^{5m}}$

Input: $A = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4$ and $B = b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3 + b_4\alpha^4$ where $a_i, b_i \in \mathbb{F}_{3^m}$.
Output: $C = A \cdot B$

1. $a_5 \leftarrow a_0 + a_1$; $b_5 \leftarrow b_0 + b_1$; $a_6 \leftarrow a_2 + a_3$; $b_6 \leftarrow b_2 + b_3$
2. $a_7 \leftarrow a_2 - a_3$; $b_7 \leftarrow b_2 - b_3$; $a_8 \leftarrow a_0 + a_4$; $b_8 \leftarrow b_0 + b_4$
3. $a_9 \leftarrow a_4 + a_5$; $b_9 \leftarrow b_4 + b_5$
4. $p_0 \leftarrow a_0 \cdot b_0$; $p_1 \leftarrow a_1 \cdot b_1$; $p_2 \leftarrow a_4 \cdot b_4$; $p_3 \leftarrow a_5 \cdot b_5$
5. $p_4 \leftarrow (a_1 - a_3) \cdot (b_1 - b_3)$; $p_5 \leftarrow (a_1 - a_6) \cdot (b_1 - b_6)$
6. $p_6 \leftarrow (a_2 - a_8) \cdot (b_2 - b_8)$; $p_7 \leftarrow (a_6 + a_9) \cdot (b_6 + b_9)$
7. $p_8 \leftarrow (a_6 - a_9) \cdot (b_6 - b_9)$; $p_9 \leftarrow (a_0 - a_4 + a_7) \cdot (b_0 - b_4 + b_7)$
8. $p_{10} \leftarrow (a_1 - a_7 - a_8) \cdot (b_1 - b_7 - b_9)$; $p_{11} \leftarrow (a_3 - a_4 + a_5) \cdot (b_3 - b_4 + b_5)$
9. $t_0 \leftarrow p_1 - p_3$; $t_1 \leftarrow p_7 + p_{10}$; $t_2 \leftarrow p_7 - p_{10}$
10. $t_3 \leftarrow p_2 + p_4$; $t_4 \leftarrow p_8 - p_6$; $t_5 \leftarrow p_{11} - t_0$
11. $c_0 \leftarrow t_4 - t_0 - t_2 - p_4$; $c_1 \leftarrow p_2 - p_0 - p_9 - t_1 - t_5$
12. $c_2 \leftarrow p_5 - p_8 - t_3 - t_5$; $c_3 \leftarrow t_2 - t_3 + t_4$
13. $c_4 \leftarrow p_4 - p_0 - p_6 + t_1$
14. **return** $C = c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + c_4\alpha^4$

B ℓ-torsion of presented curves

We provide in this appendix the largest factor ℓ of the cardinals of the curves used in Table 1 and the one with 192-bit security level mentioned in the conclusion.

B.1 Characteristic 2

- $E_{2,1}(\mathbb{F}_{2^{1117}})$:
  $\ell = 619074192321273307277438691119233058790820634893360057193377\backslash$
  $122275541424570658263412019435765493074195820297417376747932\backslash$
  $248094264569966237629582261870883925353443145570692187335548\backslash$
  $683837515601099459860669193973764482753436531478745981766945\backslash$
  $411504253650899252459234448440440995323058733022537477753547\backslash$
  $54333152682491155152733 3$ (1076 bits)
- $E_{2,1}(\mathbb{F}_{2^{3\cdot 367}})$:
  $\ell = 969479603278186730289503541042886691666123364558568701313813\backslash$
  $359745290668184127412771736661726167918225502828978334069274\backslash$
  $912792960814346728226407067755389529068277515573173949951347\backslash$
  $909708753667984651964976566357$ (698 bits)
- $E_{2,1}(\mathbb{F}_{2^{5\cdot 227}})$:
  $\ell = 3878075661272576523266141889738205178548868805960713653405 79\backslash$
  $081203820130327954203519023617159911950745211425564080964614\backslash$
  $422638824727652925391132558155138126471653299809062407679361\backslash$
  $061770177699935019248671813349799082261 81$ (733 bits)
- $E_{2,1}(\mathbb{F}_{2^{7\cdot 163}})$:
  $\ell = 527333423657076418735771198490742135902593988840498282835698\backslash$
  $225469081551234017413009393924349615391057883025970143409298\backslash$
  $825525377011095738875173885561559802962759260228479765262681\backslash$
  $163798303770288935761449011655320206737417561 01$ (753 bits)
- $E_{2,1}(\mathbb{F}_{2^{9\cdot 127}})$:
  $\ell = 345428763515337455320513346595838457562152586937282502188431\backslash$
  $308746562321219499017628832563019820634682827019770799089846\backslash$
  $906371734583958492969933493$ (487 bits)
- $E_{2,1}(\mathbb{F}_{2^{11\cdot 103}})$:
  $\ell = 455093218131231753950603619629047497847974430881584114342263\backslash$
  $233539580092770726678766917392702585626625835978899843075414\backslash$
  $305198878050459905535835655050847093202783299430021075356835\backslash$
  $014745535421646570085100803038555197229274765925231278728563\backslash$
  $736105101506288598473920529588514915 21$ (922 bits)
- $E_{2,0}(\mathbb{F}_{2^{13\cdot 89}})$:
  $\ell = 211802332252682334826944758481503093655177752110895515656983\backslash$
  $707753921755157758466447995474127577060935924385455703016958\backslash$
  $758137983194279952499146720659285364563684009302850259666276\backslash$
  $195641855840764518419327179431718554254174581321083982947322\backslash$
  $981528804011230947625432051866970535921099135184332412995164\backslash$
  093370870167293 (1044 bits)

- $E_{2,0}(\mathbb{F}_{2^{15\cdot 73}})$:
  $\ell = 980057908630054849590982407802589457588999793562532628176670\backslash$
  $097785129845409864188659712165658132978663850242420453783649\backslash$
  $646693497098428583250379810 1$ (492 bits)

B.2 Characteristic 3

- $E_{3,1}(\mathbb{F}_{3^{503}})$:
  $\ell = 545523657676112447260904563578912738373307867219686215849632\backslash$
  $469801471112426878939776725222290437653718473962733760874627\backslash$
  $315930933126581248465899651120481066111839081575164964589811\backslash$
  $985885719017214938514563804313$ (697 bits)
- $E_{3,-1}(\mathbb{F}_{3^{5\cdot 97}})$:
  $\ell = 588732453011753508013694503091100490261928459157514647309296\backslash$
  $941697666832507096172127852923090672844101$ (338 bits)
- $E_{3,-1}(\mathbb{F}_{3^{7\cdot 67}})$:
  $\ell = 198877173206052894812635074296157932741332034424157038650419\backslash$
  $395854116075771637066167599499150109958221999967622358900215\backslash$
  $860067276223554731852424604942067727364565534943781395876039\backslash$
  31839 (612 bits)
- $E_{3,-1}(\mathbb{F}_{3^{11\cdot 53}})$:
  $\ell = 211481819493746086782493570063344144137025400858064707053988\backslash$
  $970007297523680247046853068625411347010037987354570096140286\backslash$
  $131167233246014189282737634759057772983792414550023605401048\backslash$
  42866391856375045279587 (672 bits)
- $E_{3,1}(\mathbb{F}_{3^{13\cdot 43}})$:
  $\ell = 780805131216013322135909117531963030525104533454379292999972\backslash$
  $285176605028947704178161317765629689154271408914071417540782\backslash$
  $065065391844801376440344497422353298680259050077190192414921\backslash$
  $27880097531140775099199845730350794443332650516869$ (764 bits)
- $E_{3,1}(\mathbb{F}_{3^{17\cdot 67}})$:
  $\ell = 580807251443580749015157892103879774263203646148186545733767\backslash$
  $301327964005061856781013661205024478877326019301235549238196\backslash$
  $694127664203363097646101448800184666886885534005618344906757\backslash$
  $470472679514154608265809419697947368576581275390419495027194\backslash$
  $917502298809542558562159519788829324206159582134293407859461\backslash$
  $864900191066023449574873333303427818411171268971389507316485\backslash$
  $836697413405268401148957441805917547876956854138626736036981\backslash$
  $306196286962102571430752380788971186616714875260124599797020\backslash$
  43146595818195017 (1650 bits)