Cocks–Pinch curves of embedding degrees five to eight and optimal ate pairing computation

Aurore Guillevic[1], Simon Masson[1,2], and Emmanuel Thomé[1]

[1] Université de Lorraine, CNRS, Inria, Nancy, France
[2] Thales, Gennevilliers, France

Abstract. Recent algorithmic improvements of discrete logarithm computation in special extension fields threaten the security of pairing-friendly curves used in practice. A possible answer to this delicate situation is to propose alternative curves that are immune to these attacks, without compromising the efficiency of the pairing computation too much. We follow this direction, and focus on embedding degrees 5 to 8; we extend the Cocks–Pinch algorithm to obtain pairing-friendly curves with an efficient ate pairing. We carefully select our curve parameters so as to thwart possible attacks by "special" or "tower" Number Field Sieve algorithms. We target a 128-bit security level, and back this security claim by time estimates for the DLP computation. We also compare the efficiency of the optimal ate pairing computation on these curves to $k = 12$ curves (Barreto–Naehrig, Barreto–Lynn–Scott), $k = 16$ curves (Kachisa–Schaefer–Scott) and $k = 1$ curves (Chatterjee–Menezes–Rodríguez-Henríquez).

1 Introduction

Constructive pairings have been introduced in the cryptographic world in the 2000's with a one round tripartite Diffie–Hellman key exchange [31], identity-based encryption [12], and short signatures [13]. More recently, new applications have been proposed, e.g. zero-knowledge proofs [11] used in the Zcash cryptocurrency and electronic voting.

Pairing-based cryptography relies on the hardness of the discrete logarithm (DL) problem over two groups: an elliptic curve $E(\mathbb{F}_p)$ and a finite field $\mathbb{F}_{p^k}$, k being the *embedding degree*. Advances on discrete logarithm computation over special field extensions $\mathbb{F}_{p^k}$ force us to review not only the parameter sizes of curves used in practice, but also the families of curves used to generate parameters.

The computation of discrete logarithms in finite fields and the factorisation of large integers are both addressed by the Number Field Sieve algorithm (NFS). Its complexity is $L_{p^k}(1/3, c + o(1))$, with the notation $L_{p^k}(\alpha, c) = \exp(c(\log p^k)^\alpha(\log\log p^k)^{1-\alpha})$. For the families of finite fields that we consider in this paper, we have $1.526 \leq c \leq 2.20$. The value of c depends on the variant of the NFS algorithm. For prime fields $\mathbb{F}_p$, $c = 1.923$ and this complexity was considered as the reference for choosing the size of *any* finite field $\mathbb{F}_{p^k}$, where p is medium to large compared to p^k. However, for some degrees k the NFS algorithm can be parameterised differently, yielding a better complexity with a smaller c. The size of the finite field must then be increased so as to maintain the same security as previously thought, that is with $c = 1.923$. In the context of finite fields (unrelated to pairings), so-called *special* primes p are subject to the *special* NFS (SNFS) variant with $c = 1.526$: in these cases, the size of p gives a false sense of security [28,51,48,46,27]. The most efficient pairings were obtained with specific families of pairing-friendly curves, where the prime p is special (see Table 2). In 2013, Joux and Pierrot exploited this weakness [32]. In 2015, Barbulescu, Gaudry and Kleinjung revisited Schirokauer's Tower NFS (TNFS) variant, and obtained $c = 1.526$ in a theoretical "special Tower NFS" variant (STNFS). In 2016, Kim and Barbulescu applied the STNFS variant to finite fields of composite extension degree k and obtained, in the best case, an algorithm of complexity with $c = 1.526$ [35]. This means that asymptotically, the size of the finite field $\mathbb{F}_{p^k}$ should be doubled to provide the same security as thought before.

However, the asymptotic complexity does not provide enough accuracy to deduce the sizes of the finite fields that we would like to use for cryptography today. Menezes, Sarkar and Singh already showed that the efficiency of the STNFS variants depends on the total size and the extension degree [41], and that the STNFS variant with the best asymptotic complexity is not necessarily the best variant that applies to a 3000-bit finite field. Then Barbulescu and Duquesne proposed a way to refine the estimates [7] and proposed parameters for 128 bits of security for BN, BLS, and KSS curves (where $k = 12$, 16 and 18).

The rule of thumb that prime fields of 3072 to 3200 bits offer 128 bits of security is extrapolated from the asymptotic complexity, rescaled according to a record computation. It almost systematically relies on the bodacious assumption $o(1) = 0$ in the asymptotic formula. Since a record computation is not available for the TNFS and STNFS algorithms, the papers [41,7] simulate a simplified version of the new algorithms to estimate their cost. We recall in Table 1 the popular pairing-friendly curves before the STNFS algorithm (2015), and the new propositions of [7], together with our estimate of the running-time of a DL computation in the corresponding fields $\mathbb{F}_{p^k}$.

parameters						Cost of DL computation		
curve	$\log_2 p$	$\log_2 r$	k	$\log_2 p^k$	$\deg p(u)$	DL on $E(\mathbb{F}_p)$ ($\approx \sqrt{r}$)	STNFS on $\mathbb{F}_{p^k}$ [7]	§ B
BN	254	254	12	3039	4	2^{127}	2^{100}	$\mathbf{2^{103}}$
BN	446	446	12	5343	4	2^{223}	–	$\mathbf{2^{132}}$
BN	462	462	12	5535	4	2^{231}	2^{131}	$\mathbf{2^{134}}$
BLS12	381	255	12	4572	6	2^{127}	–	$\mathbf{2^{126}}$
BLS12	461	308	12	5525	6	2^{154}	2^{132}	$\mathbf{2^{134}}$
KSS16	330	257	16	5280	10	2^{128}	2^{139}	$\mathbf{2^{141}}$
KSS16	339	263	16	5411	10	2^{131}	2^{139}	$\mathbf{2^{141}}$

Table 1: Sizes and DL cost estimates.

BN, $\rho = 1$, $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$	
BN-254	$u = -2^{62} - 2^{55} - 1$ (2010) [44]
BN-446	$u = 2^{110} + 2^{36} + 1$ (2010) [44]
BN-462	$u = 2^{114} + 2^{101} - 2^{14} - 1$ (2018) [7]
BLS12, $\rho = 3/2$, $p = (u^6 - 2u^5 + 2u^3 + u + 1)/3$	
BLS12-381	$u = -(2^{63} + 2^{62} + 2^{60} + 2^{57} + 2^{48} + 2^{16})$ (2017 Zcash [15])
BLS12-461	$u = -2^{77} + 2^{50} + 2^{33}$ (2018) [7]
KSS16, $\rho = 5/4$	
$p = (u^{10} + 2u^9 + 5u^8 + 48u^6 + 152u^5 + 240u^4 + 625u^2 + 2398u + 3125)/980$	
KSS16-330	$u = -2^{34} + 2^{27} - 2^{23} + 2^{20} - 2^{11} + 1$ (2018) [7]
KSS16-339	$u = 2^{35} - 2^{32} - 2^{18} + 2^8 + 1$ (2018) [7]

Table 2: Parameters of commonly used pairing-friendly curves

Generation of pairing-friendly curves. Pairings on elliptic curves map pairs in $E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})[r]$ to $\mathbb{F}_{p^k}$, and the *embedding degree* k should be small, say $1 \le k \le 20$. The first suggested pairings used supersingular curves, because they were the first known way to produce curves with a small embedding degree. Then two other approaches were proposed (and surveyed in [26]). On the one hand, *polynomial* methods parameterise the characteristic p, the trace t, and the curve order r by polynomials. The parameterisation enables two fast variants of the Tate pairing: the ate or optimal ate pairing [50]. A Tate pairing computation contains an internal loop (the Miller loop) of length $\log_2 r$, while this length is $\log_2(t-1)$ (resp. $(\log_2 r)/\varphi(k)$) for an ate (resp. optimal ate) pairing. On the other hand, non-polynomial methods were also proposed. Because of the large gap in pairing efficiency, the latter methods attracted less attention, and were not optimised.

Pairing-friendly curves from polynomial constructions enjoy fast implementations (e.g., [2]). However, the downside is that since p is parameterised by a polynomial, the Kim–Barbulescu STNFS algorithm applies (at least in some cases), and security estimates need to be revised. Alternatively, cryptographers look for new pairing-friendly curves. Chatterjee, Menezes and Rodríguez-Henríquez propose in [16] to revisit curves of degree one (and trace $t = 2$), avoiding the TNFS attack. The target finite field is a prime field so it only requires $\log_2(p) \ge 3072$ to get 128-bits of security [43]. Fotiadis and Konstantinou propose in [24] families of elliptic curves from a variant of the (polynomial) Brezing–Weng method. For composite embedding degree, they increase the parameter sizes in order to get TNFS-resistant curves, but they also propose curves of prime embedding degree for which the TNFS attack is restricted to $\deg h = k$ only. Unfortunately, the prime p has a polynomial form so the special variants (SNFS and STNFS) still apply.

Our contribution. In this article, we take a different approach: to avoid having to increase the size of the target finite field $\mathbb{F}_{p^k}$, we choose a Cocks–Pinch curve so that p is not special and the special variants (SNFS, STNFS) do not apply.

But on Cocks–Pinch curves, no optimal ate pairing is available, and the ate pairing is as slow as the Tate pairing because of a large trace t. So we modify the Cocks–Pinch method to obtain *a trace of smallest possible size* $(\log_2 r)/\varphi(k)$, and arrange so that the ate pairing or its variant [50, §2.2] is available. We obtain an optimal pairing as defined by Vercauteren in [50]. We generate curves of embedding degree 5 to 8 in order to compare the efficiency of the ate pairing with different sizes of parameters.

- For composite extension degrees ($k = 6$ and 8), we reuse some of the optimisations from the literature to obtain a pairing computation that is as efficient as with competing constructions. While it is true that by doing so, we endow our prime p with some special structure, we argue in this paper that the multivariate nature of our parameterisation offers much more flexibility than known constructions, and to our knowledge thwarts all known "special" variants of NFS.
- For prime embedding degree ($k = 5$ and 7), the TNFS attack is restricted to one choice: $\deg h = k$. It leads to a smaller target finite field than in the composite cases. But a prime embedding degree eliminates some optimisation opportunities for the pairing computation.

This article also gives cost estimates and comparisons based on existing software. We show that the added confidence in the DL hardness can be obtained without sacrificing the pairing efficiency too much.

Organisation of the paper. In Section 2, we recall the Tate and ate pairings, in particular the notions of Miller loop and final exponentiation. We present in Section 3 our Cocks–Pinch variant to construct secure curves with an efficient pairing. Section 4 addresses the cost estimates for DL computation with the known variants of the NFS algorithm. In Section 5 we provide parameters of curves for 128 bits of security together with the analysis of the pairing cost. We compare the pairing efficiency to their challengers: BN, BLS12 and KSS16 curves, and embedding degree one curves from [16].

1.1 Code repository

Companion code is provided for several sections in this article, including code to reproduce experimental data. The code repository is publicly accessible at:

`https://gitlab.inria.fr/smasson/cocks-pinch-variant.git`.

2 Background on pairings

We present here the computation of two pairings used in practice, the Tate and ate pairings. Then we list refinements in the case of ate pairing on BN curves.

Let E be an elliptic curve defined over $\mathbb{F}_p$. Let π_p be the Frobenius endomorphism $(x, y) \mapsto (x^p, y^p)$. Its minimal polynomial is $X^2 - tX + p$ and t is called the *trace*. Let r be a prime divisor of $\#E(\mathbb{F}_p) = p + 1 - t$. The r-torsion subgroup of E is noted $E[r] := \{P \in E(\overline{\mathbb{F}_p}), [r]P = \mathcal{O}\}$ and has two subgroups of order r (eigenspaces of π_p in $E[r]$) that are useful for pairing applications: $\mathbb{G}_1 = E[r] \cap \ker(\pi_p - [1])$ and $\mathbb{G}_2 = E[r] \cap \ker(\pi_p - [p])$. The latter is defined over $\mathbb{F}_{p^k}$, where the *embedding degree* k is the smallest integer $k \in \mathbb{N}^*$ such that r divides $p^k - 1$. A pairing-friendly curve is an elliptic curve that satisfies the following conditions: p and r are prime numbers, t is relatively prime to p, and k should be small. The discriminant $-D$ is the fundamental discriminant of the quadratic imaginary field defined by $X^2 - tX + p$ (so that $t^2 - 4p = -Dy^2$ for an integer y). All constructions require that $|D|$ be small enough, so that the complex multiplication method is feasible (the record computation in [49] has $|D| \sim 10^{16}$). The ρ-value of E is defined by $\rho(E) = \log(p)/\log(r)$. The "ideal" case is $\rho(E) \approx 1$ when $r = \#E(\mathbb{F}_p)$.

We recall the Tate and ate pairings definition, based on the same two steps: evaluating a function $f_{s,Q}$ at a point P, and then raising it to the power $(p^k - 1)/r$. (Sometimes the pairing is said *reduced* to stress the final exponentiation). The function $f_{s,Q}$ has divisor $\mathrm{div}(f_{s,Q}) = s(Q) - ([s]Q) - (s-1)(\mathcal{O})$ and satisfies

$$f_{i+j,Q} = f_{i,Q} f_{j,Q} \frac{\ell_{[i]Q,[j]Q}}{v_{[i+j]Q}}$$

where $\ell_{[i]Q,[j]Q}$ and $v_{[i+j]Q}$ are the two lines needed to compute $[i+j]Q$ from $[i]Q$ and $[j]Q$ (ℓ through the two points, v the vertical). We compute $f_{s,Q}(P)$ with the Miller loop presented in Algorithm 1.

Algorithm 1: MillerLoop(s, P, Q) – Compute $m = f_{s,Q}(P)$.

$m \leftarrow 1$; $S \leftarrow Q$
for *b from the second most significant bit of s to the least* **do**
  $m \leftarrow m^2 \cdot \ell_{S,S}(P)/v_{[2]S}(P)$; $S \leftarrow [2]S$
  if $b = 1$ **then**
    $m \leftarrow m \cdot \ell_{S,Q}(P)/v_{S+Q}(P)$; $S \leftarrow S + Q$
return m

The Tate and ate pairings are defined by

$$\mathrm{Tate}(P,Q) := f_{r,P}(Q)^{(p^k-1)/r}, \text{ and } \mathrm{ate}(P,Q) := f_{t-1,Q}(P)^{(p^k-1)/r}$$

where $P \in \mathbb{G}_1 \subset E[r](\mathbb{F}_p)$ and $Q \in \mathbb{G}_2 \subset E[r](\mathbb{F}_{p^k})$. The values $\mathrm{Tate}(P,Q)$, and $\mathrm{ate}(P,Q)$ are in the "target" group $\mathbb{G}_T$ of r-th roots of unity in $\mathbb{F}_{p^k}$.

Before we analyse in Section 5 the cost of computing pairings, we briefly comment on the CM discriminant $-D$. When $D = 3$ (resp. $D = 4$), the curve has complex multiplication by $\mathbb{Q}(j)$ (resp. $\mathbb{Q}(i)$), so that a twist of degree 6 (resp. 4) exists. When E has d-th order twists for some $d|k$, then $E[r](\mathbb{F}_{p^k})$ is isomorphic to $E'[r](\mathbb{F}_{p^{k/d}})$ for some twist E'. Dealing with the latter is easier. Therefore, composite extension degrees are often an invitation to choose $D = 3$ or $D = 4$.

3 Construction of secure curves with efficient ate pairing

In this section, we look for curves that are not threatened by recent variants of NFS. We make the following observations.

- All families of curves in [26] compute p as a polynomial evaluated at a chosen integer. This (often) enables the STNFS algorithm [32], so that the DL problem in $\mathbb{F}_{p^k}$ is easier than in other fields of same bit length.
- While composite extension degrees are appealing for fast pairing computation (see §2), they also offer additional parameterisation choices for the TNFS algorithm [35]. This also makes DL computations in $\mathbb{F}_{p^k}$ more efficient.

We wish to avoid special primes. Furthermore, as our range of interest $5 \le k \le 8$ contains the composite degrees $k = 6$ and $k = 8$, we acknowledge the need to choose the size of p so as to compensate the TNFS attack.

Algorithm 2: ModifiedCocksPinch$(k, -D, T_0, T_{\max}, \lambda_r, \lambda_p)$ – Compute a pairing-friendly curve of embedding degree k and fundamental discriminant $-D$, where $\lceil \log_2(p) \rceil = \lambda_p$ and $\lceil \log_2(r) \rceil = \lambda_r$.

for $T \in \{T_0, \ldots, T_{max}\}$ **do**
  if $r = \Phi_k(T)$ *is not prime* **then continue**
  if $\lceil \log_2(r) \rceil \neq \lambda_r$ *or* $-D$ *is not a square mod* r **then continue**
  for i *in* $\{1, 2, \ldots, k-1\}$ *such that* $\gcd(i, k) = 1$ **do**
    $t_0 = T^i + 1 \mod r$; $y_0 = (t_0 - 2)/\sqrt{-D} \mod r$ ▷ centered representatives
    Let $\pi_0 = \frac{t_0 + y_0\sqrt{-D}}{2}$.
    Choose h_t and h_y such that $\pi = \pi_0 + \frac{h_t + h_y\sqrt{-D}}{2} r$ is an algebraic integer, and $\lceil \log_2(\pi\bar{\pi}) \rceil = \lambda_p$.
    $t = t_0 + h_t r$; $y = y_0 + h_y r$; $p = \pi\bar{\pi} = (t^2 + Dy^2)/4$
    if $p \equiv 1 \mod k$ **then**   ▷ optimisation; see Remark 2
      if p *is prime* **then return** $[p, r, T, t, y]$

To avoid special primes, we revisit the Cocks–Pinch method, which constructs pairing-friendly curves with freely chosen embedding degree k and discriminant $-D$. The classical Cocks–Pinch algorithm first fixes the prime r and deduces a root of unity mod r to compute t and then p satisfying the conditions of pairing-friendly curves. Instead, we first choose T small, and then compute r such that T is a k-th root of unity mod r. Then we observe that $f_{T,Q}(P)$ like $f_{t-1,Q}(P)$ gives a Miller loop of a bilinear pairing.

Our variant is given in Algorithm 2. The trace $\bar{t} \in \mathbb{Z}/r\mathbb{Z}$ can be any of $\bar{t} = T^i + 1 \bmod r$ where $\gcd(i, k) = 1$. Then $\bar{y} = (\bar{t} - 2)/\sqrt{-D} \bmod r$ as in the original method. We then lift $\bar{t}, \bar{y}$ to $t_0, y_0 \in \mathbb{Z}$. The choice of the cofactors h_t and h_y in Algorithm 2 must abide by certain rules so that the Weil number π is an algebraic integer: if $-D \equiv 0 \mod 4$, then $t_0 + h_t$ must be even, and if $-D \equiv 1 \mod 4$, then $t_0 + y_0 + h_t + h_y$ must be even. For p to have the desired bit length, we notice that h_t and h_y must be chosen in an annulus-like region given by the equation $2^{\lambda_p+1} \leq (t_0 + h_t r)^2 + D(y_0 + h_y r)^2 < 2^{\lambda_p+2}$.

The Miller algorithm in the ate pairing iterates on $t - 1 = T^i$, a k-th root mod r. Iterating on another root of unity also gives a pairing, as remarked by the following statement.

Theorem 1 ([50, §2.2] and the references therein). *Let $P \in \mathbb{G}_1 = E[r] \cap \ker(\pi_p - [1])$ and $Q \in \mathbb{G}_2 = E[r] \cap \ker(\pi_p - [p])$. Let T be a k-th root of unity mod r. Then, $f_{T,Q}(P)$ defines a bilinear pairing and*

$$\mathit{Tate}(Q,P)^L = f_{T,Q}(P)^{c(p^k-1)/N}$$

where $N = \gcd(T^k - 1, p^k - 1)$, $T^k - 1 = LN$, and $c = \sum_{i=0}^{k-1} T^{k-1-i} p^{ji}$.

In particular, our T in Algorithm 2 is convenient and defines an optimal ate pairing in the sense of [50, Definition 1]:

$$\mathrm{OptimalAte}(P,Q) := f_{T,Q}(P)^{\frac{p^k-1}{r}} .$$

Does Algorithm 2 produce primes of a special form? In this paper, we will discuss in particular whether we hold to our promise that p, as issued by Algorithm 2, is not special. The prime p has the form

$$p = \frac{1}{4}\left((t_0 + h_t r)^2 + D\,(y_0 + h_y r)^2\right)$$

where t_0, y_0 are centered representatives of $T^i + 1 \bmod r$ and $(t_0 - 2)/\sqrt{-D} \bmod r$ resp., and both r and t_0 are low-degree polynomials in T.

If T, h_t, and h_y are chosen by Algorithm 2 as random integers of the desired bit length, and that D is arbitrary (then y_0 has no nice sparse polynomial expression in T), then the expression above is considered unlikely to yield any computational advantage to an attacker.

On the other hand, efficiency considerations may lead us to choose D specially, so as to allow extra automorphisms on the curve, for example choose $-D$ as the discriminant of $\mathbb{Q}(\zeta_d)$ for some $d \mid k$. Then $\sqrt{-D}$ typically has a low-degree polynomial expression in T. If T, h_t, and h_y are then chosen with low Hamming weight, the answer is less clear. In comparison to other pairing-based constructions however (see Table 2), we have here a *multivariate* expression for p (it depends on T, h_t, and h_y). There exists no special-purpose NFS construction that adapts well to this situation. It seems hard, in particular, to derive a univariate expression that enjoys the properties of sparsity that are desired for NFS polynomial selection.

As an illustration, here is the multivariate expression of p in the case $k = 8$, $D = 4$, and $i = 5$.

$$\begin{aligned} p = {} & (h_t^2 + 4h_y^2)T^8 - 4h_y T^7 - (4h_y - 1)T^6 - 2(h_t - 1)T^5 \\ & + (2h_t^2 + 8h_y^2 + 2h_t + 1)T^4 - 4h_y T^3 - (4h_y - 1)T^2 - 2(h_t + 1)T \\ & + (h_t^2 + 4h_y^2 + 2h_t + 1). \end{aligned}$$

4 DL cost estimate and size requirements

In this section, we would like to determine, for each embedding degree $k \in \{5,6,7,8\}$, the appropriate bit length of p so that the pairings on the curves constructed in §3 match the 128-bit security level. This leads us to assess the hardness of the DL computation in the subgroup of (256-bit) prime order r of $\mathbb{F}_{p^k}$, In terms of notations, we naturally search for different values of p for each k, so that p depends on k. For brevity however, we prefer to keep the notation p rather than use p_k.

We also do the same analysis for other curve families. We strive to take into account in our analysis the different known NFS variants. This complements the study in [7].

4.1 Strategy for estimating NFS cost

Estimating the computational cost of the number field sieve is a delicate task because of the inherent complexity of NFS, its numerous parameters and its variants, and moreover because of the very different nature of the different steps of the algorithm. This is a vastly different situation from, say, the assessment of the DLP hardness for elliptic curves, where the lack of any algorithm more advanced than $O(\sqrt{r})$ algorithms makes estimations comparatively easy.

The two main steps of NFS are relation collection and linear algebra. We tried to estimate both, in terms of "elementary operations" that combine a memory access (to sieve array elements, to vector coefficients) as well as arithmetic operations. Our simulation methodology is not dissimilar to the one used in [41,7]: starting from an NFS or TNFS setup, we estimate the norms involved in the relation collection step, and the associated smoothness probability. Further detail is given in Appendix B.2. We summarise here the variations that we introduce:

- In the NFS context, coprime (a,b) pairs that form the primary source of candidates for smoothness are counted as the total number of pairs in the sieve area times the factor $\frac{6}{\pi^2} = \frac{1}{\zeta(2)}$. In the TNFS context, the analogue of this scaling is given by $\frac{1}{\zeta_K(2)}$ where ζ_K is the Dedekind zeta function of the base number field [30].
- We compute the smoothness probabilities of the two norms that are deduced from (a,b) pairs as the average of the smoothness probability of norms over 10^6 samples, instead of the smoothness probability of the average norm over 25600 samples.
- Estimating the matrix size that results from a given set of parameter choices requires to estimate the reduction factor of the so-called filtering step. As we show in Appendix B.2, recent large computations chose parameters very differently, which led to vastly different reduction factors. Based on a rationale for parameter choice that is in accordance with previous computation with the `cado-nfs` software, we estimate the filtering step as providing a (conservative) constant reduction factor of 9. This is very different from the reasoning in [7] and we justify this choice in Appendix B.2.

Original family	curve or field	k	$\log_2 p^k$	$\log_2 r$	dh	df,dg	poly	cost
prime fields	Oakley p	1	3072	3071	–	(5,4)	JL	2^{127}
	prime field	1	3200	256	–	(5,4)	JL	2^{128}
[16]		1	3072	256	–	(5,4)	JL	2^{128}
This work	$k5$, $p663$	5	3318	256	5	(6,5)	JL	2^{128}
	$k6$, $p672$, $D = 3$	6	4032	256	2	(6,3)	Conj	2^{128}
	$k7$, $p512$	7	3584	256	7	(6,5)	JL	2^{132}
	$k8$, $p544$, $D = 1$	8	4349	256	2	(8,4)	Conj	2^{130}
[26, Ex. 6.8]	BN-462	12	5534	462	6	(8,2)	JP	2^{135}
[26, §6.1]	BLS12-381	12	4572	255	6	(12,2)	JP	2^{126}
	BLS12-461	12	5525	256	6	(12,2)	JP	2^{134}
[26, Ex. 6.11]	KSS16-330	16	5280	257	16	(10,1)	JP	2^{141}
	KSS16-339	16	5411	263	16	(10,1)	JP	2^{141}

Table 3: Comparison of DL cost estimates. Polynomial selection methods are abbreviated as Conj for the conjugation method [8], JL for Joux–Lercier, JP for the Joux–Pierrot STNFS variant (when it improves on JL).

4.2 Extension degrees 5 to 8, and comparison

Our simulation results are given in Table 3, which covers both the method in §3 as well as competing curves. For reference, we also include a cost estimate, obtained with the same method, for 3072-bit prime fields. This fits reasonably well with the commonly accepted idea that this size matches 128-bit security (see e.g. [43, §5.6]). We compared the cost estimates using the special form of p to generic settings (such as the conjugation method).

It follows from Table 3 that for curves in §3, the "special" algorithms such as STNFS offer no computational advantage. In other words, p is *not special*, at least not in an exploitable way, for the sizes we consider.

4.3 Evolution of DL cost as $\log_2 p$ changes

As an additional measure of the inapplicability of special algorithms to the constructions in §3, we also study the evolution of the DL computation cost as the size of p varies. We do this for $k = 6$, assuming $D = 3$, and $k = 8$ with $D = 4$. Figure 4 compares prime-order MNT curves ($k = 6$, p is not special) and our curves.

It follows from Figure 4 that the special form of p for our curves does offer an advantage compared to the generic Conjugation–TNFS algorithm for very small sizes of p, between 512 and 560 bits for $k = 6$, $D = 3$ (this corresponds to $|h_t|$ and $|h_y|$ of at most 20 bits), and between 512 and 536 bits for $k = 8$, $D = 4$ ($|h_t|$ and $|h_y|$ of at most 12 bits). For larger parameters, the generic TNFS algorithm is faster.

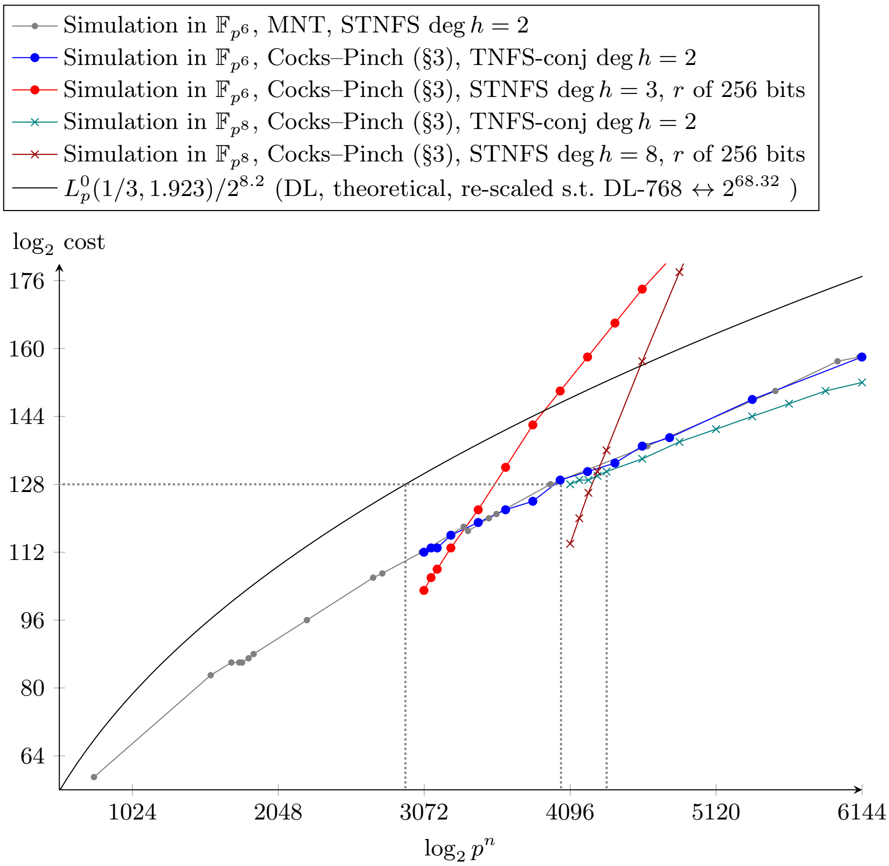

Fig. 4: Cost estimates in $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^8}$, comparison of MNT curves and curves of §3 for $k = 6$ and $k = 8$. The degree of h was chosen to minimize the estimated cost of (S)TNFS. The notation L_N^0 stresses the fact that the asymptotic complexity is used without $o(1)$.

5 Pairing cost

We now count the number of operations over $\mathbb{F}_p$ to compute the optimal ate pairing with Algorithms 3, 4 and 5. We denote by $\mathbf{m}_k$, $\mathbf{s}_k$, $\mathbf{i}_k$ and $\mathbf{f}_k$ the costs of multiplication, squaring, inversion, and p-th power Frobenius in $\mathbb{F}_{p^k}$, and by $\mathbf{m} = \mathbf{m}_1$ the multiplication in $\mathbb{F}_p$. We neglect additions and multiplications by small constants. In order to provide a common comparison base, we give theoretical costs for $\mathbf{m}_k$ and $\mathbf{s}_k$ using Karatsuba-like formulas [22,42,19]. Inversions are computed using the expression below, which relies on efficient Frobenius powers:

$$a^{-1} = (\mathrm{Norm}_{\mathbb{F}_q^k/\mathbb{F}_q}(a))^{-1} \times a^q \times \cdots \times a^{q^{k-1}}.$$

Remark 2 (Frobenius cost). For the latter to perform well, it is very useful to have $p \equiv 1 \bmod k$ (as we did in Algorithm 2), and define $\mathbb{F}_{p^k}$ as $\mathbb{F}_{p^k} = \mathbb{F}_p[x]/(x^k - \alpha)$: let $w = \sum_{i=0}^{k-1} \omega_i x^i \in \mathbb{F}_{p^k} = \mathbb{F}_p[x]/(x^k - \alpha)$. Then, $w^{p^j} = \omega_0 + \sum_{i=1}^{k-1} \omega_i x^{ip^j}$. The terms x^{ip^j} do not depend on w and are precomputed. By Euclidean division by k, $x^{ip^j} = x^{u_j k + i} = \alpha^{u_j} x^i$. Therefore we have at most $\mathbf{f}_k = (k-1)\mathbf{m}$ for any p^j-th power Frobenius. Note that for k even, we have $x^{k/2\cdot(p^j-1)} = \alpha^{(p^j-1)/2} = \pm 1$ so that $x^{k/2\cdot p^j} = \pm x^{k/2}$, whence one multiplication can be saved.

Consequences of the above are given in [47], notably $\mathbf{i}_2 = 2\mathbf{m} + 2\mathbf{s}_1 + \mathbf{i}_1$ and $\mathbf{i}_3 = 9\mathbf{m} + 3\mathbf{s}_1 + \mathbf{i}_1$, neglecting additions. Recursive application yields $\mathbf{i}_k$ for $k = 2, 3, 4, 6, 8, 12, 16$. For $k = 5$ we use $a^q \times \cdots \times a^{q^4} = ((a^q)^{1+q})^{1+q^2}$ so that $\mathbf{i}_5 = 4\mathbf{f}_5 + 2\mathbf{m}_k + \mathbf{i}_1 + 10\mathbf{m}$, and $\mathbf{i}_7$ is obtained in a similar way. Table 5 summarises these theoretical costs, and includes specialised costs for cyclotomic squares (see [29, §3.1]). We compared Table 5 with timings of the RELIC library [4] for primes p of 6 to 8 machine words and $k = 2, 6, 12$ on an Intel Core i5-4570 CPU, 3.20GHz. The accordance is satisfactory (within 10%), to the point that we use Table 5 as a base. Additionally, we also measured the relative costs of $\mathbf{i}_1$, $\mathbf{s}_1$, and $\mathbf{m}$ on the same platform, leading to $\mathbf{i}_1 \approx 25\mathbf{m}$ and $\mathbf{s}_1 \approx \mathbf{m}$.

k	1	2	3	5	6	7	8	12	16
$\mathbf{m}_k$	$\mathbf{m}$	$3\mathbf{m}$	$6\mathbf{m}$	$13\mathbf{m}$	$18\mathbf{m}$	$22\mathbf{m}$	$27\mathbf{m}$	$54\mathbf{m}$	$81\mathbf{m}$
$\mathbf{s}_k$	$\mathbf{m}$	$2\mathbf{m}$	$5\mathbf{m}$	$13\mathbf{m}$	$12\mathbf{m}$	$22\mathbf{m}$	$18\mathbf{m}$	$36\mathbf{m}$	$54\mathbf{m}$
$\mathbf{f}_k$	$0\mathbf{m}$	$0\mathbf{m}$	$2\mathbf{m}$	$4\mathbf{m}$	$4\mathbf{m}$	$6\mathbf{m}$	$6\mathbf{m}$	$10\mathbf{m}$	$14\mathbf{m}$
$\mathbf{s}_k^{\mathrm{cyclo}}$					$6\mathbf{m}$		$12\mathbf{m}$	$18\mathbf{m}$	$36\mathbf{m}$
$\mathbf{i}_k - \mathbf{i}_1$	0	$4\mathbf{m}$	$12\mathbf{m}$	$52\mathbf{m}$	$34\mathbf{m}$	$116\mathbf{m}$	$44\mathbf{m}$	$94\mathbf{m}$	$134\mathbf{m}$
$\mathbf{i}_k$, with $\mathbf{i}_1 = 25\mathbf{m}$	$25\mathbf{m}$	$29\mathbf{m}$	$37\mathbf{m}$	$77\mathbf{m}$	$59\mathbf{m}$	$141\mathbf{m}$	$69\mathbf{m}$	$119\mathbf{m}$	$159\mathbf{m}$

Table 5: Relative cost of $\mathbf{m}_k$, $\mathbf{s}_k$ and $\mathbf{i}_k$ for our finite field extensions.

5.1 Miller loop

The Miller loop evaluates functions defined over $\mathbb{F}_{p^k}$, at a point of $E(\mathbb{F}_p)$. Algorithm 3 is essentially a repetition of Algorithm 1 with a few modifications related to practice: it is desirable to separate numerators and denominators so as to compute only one inversion at the end. Furthermore, the argument T may conveniently be handled in binary non-adjacent form $T = \sum_{i=0}^{n} b_i 2^i = (b_n b_{n-1} \ldots b_2 b_1 b_0)_{\text{2-NAF}}$ with $b_i \in \{-1, 0, 1\}$. We use the notation $\mathrm{HW}_{\text{2-NAF}}$ for the number of non-zero b_i.

Algorithm 3 uses three helper functions that are detailed as Algorithms 4 and 5. The input point S as well as the output point $\mathbf{S}$ in these algorithms are in Jacobian coordinates [20]: the quadruple (X, Y, Z, Z^2) represents the affine point $(X/Z^2, Y/Z^3)$. This saves inversions and multiplications.

In Table 6, we give the cost of the line computations for $5 \le k \le 8$. As it turns out, the embedding degree k affects the pairing cost in multiple ways. As mentioned in §2 (see also [21,3]), twists allow efficient computations for $6|k$ (resp. $4|k$) if we set $D = 3$ (resp. $D = 4$). During Algorithm 3, the factors in proper subfields of $\mathbb{F}_{p^k}$ are neutralised during the final exponentiation, and hence are not computed. In particular, λ_d and m_d are omitted from the Miller loop computation for these curves. Algorithms 4 and 5 are also simplified[3]. Table 6 takes adaptations of these optimisations into account when twists are available[4], while the data for $k = 5$ and $k = 7$ comes straight from Algorithms 3, 4 and 5. We denote by $\mathbf{mc}_k$ the cost of multiplying a constant $c \in \mathbb{F}_p$ by an element of $\mathbb{F}_{p^k}$. A consequence of Table 6 is that the Jacobi quartic, Hessian, and Edwards models induce comparatively expensive pairing computations, and the Weierstrass model is preferred in practice.

The final cost of Algorithm 3 is given by the following formula, where the notation $\mathbf{c}_X$ denotes the cost of step X, or algorithm X.

$$\begin{aligned}\mathbf{c}_{\textsc{MillerLoop}} =& (\log_2(T) - 1)\,(\mathbf{c}_{\textsc{DoubleLine}} + \mathbf{c}_{\textsc{VerticalLine}}) \\ &+ (\log_2(T) - 2)\mathbf{c}_{\textsc{Update1}} \\ &+ (\mathrm{HW}_{\text{2-NAF}}(T) - 1)(\mathbf{c}_{\textsc{AddLine}} + \mathbf{c}_{\textsc{VerticalLine}} + \mathbf{c}_{\textsc{Update2}}) \\ &+ (\text{only if } k \in \{5, 7\})\mathbf{i}_k. \end{aligned} \tag{1}$$

5.2 Final exponentiation, first and second part

The final exponentiation to the power $(p^k - 1)/r$ is computed in two steps, named first and second part, corresponding to the two factors of the exponent: $\frac{p^k-1}{\Phi_k(p)} \times \frac{\Phi_k(p)}{r}$.

[3] In particular, the line computations involve some sparse products, e.g. $(\sum a_i x^i) \times (\sum b_i x^i)$ in $\mathbb{F}_{p^8}$ over $\mathbb{F}_{p^2}$ with $a_1 = 0$ (see [21]), which costs $8\mathbf{m}_2$ by Karatsuba. Note that [52] claims $7\mathbf{m}_2$ but with no explicit formula. We were not able to match this. The work [34, §3.3] obtained $7\mathbf{m}_2$ in favorable cases at a cost of extra precomputations.

[4] The Edwards model is not available for a quartic or sextic twist because there is no 4-torsion point on these twists, only the quadratic twist can be in Edwards form [40]. The Jacobi quartic model is not available for a cubic or sextic twist because there is no 2-torsion point on the twist. The Hessian model is compatible with cubic twists but not sextic twists.

Algorithm 3: MillerLoop($T, P \in E(\mathbb{F}_p), Q \in E(\mathbb{F}_{p^k})$) – Compute $f_{T,Q}(P)$.

$(m_n, m_d) \leftarrow (1, 1)$; $S \leftarrow Q$
for *b from the second most significant bit of* $|T|$ *to the least* **do**
  $(\lambda_n, \lambda_d) \leftarrow \ell_{S,S}(P)$; $S \leftarrow [2]S$ ▷DoubleLine
  $(\mu_n, \mu_d) \leftarrow v_S(P)$ ▷VerticalLine
  $(m_n, m_d) \leftarrow (m_n^2 \lambda_n \mu_d, m_d^2 \lambda_d \mu_n)$ ▷Update1
  if $b = \pm 1$ **then**
    $(\lambda_n, \lambda_d) \leftarrow \ell_{S,bQ}(P)$; $S \leftarrow S + bQ$ ▷AddLine
    $(\mu_n, \mu_d) \leftarrow v_S(P)$ ▷VerticalLine
    $(m_n, m_d) \leftarrow (m_n \lambda_n \mu_d, m_d \lambda_d \mu_n)$ ▷Update2
if $T < 0$ **then** $(m_n, m_d) \leftarrow (m_d, m_n)$
return m_n / m_d

Algorithm 4: AddLine(S, Q, P) and DoubleLine(S, P) – Given $S, Q \in E(\mathbb{F}_{p^k})$, compute $S + Q$ (resp. $2S$) and the evaluation of the line (SQ) (resp. the tangent at S) at $P \in E(\mathbb{F}_p)$.

AddLine	DoubleLine
$(X, Y, Z, Z_2) \leftarrow S$	$(X, Y, Z, Z_2) \leftarrow S$
$(x_Q, y_Q) \leftarrow Q$	$(x_P, y_P) \leftarrow P$
$(x_P, y_P) \leftarrow P$	$t_1 \leftarrow Y^2$
$t_1 \leftarrow x_Q \cdot Z_2 - X$	$t_2 \leftarrow 4X \cdot t_1$
$t_2 \leftarrow y_Q \cdot Z \cdot Z_2 - Y$	**if** $a = -3u^2$ *for a small* $u \in \mathbb{F}_p$ **then**
$t_3 \leftarrow t_1^2$	$t_3 \leftarrow 3(X - uZ_2) \cdot (X + uZ_2)$
$t_4 \leftarrow t_1 \cdot t_3$	**else**
$t_5 \leftarrow X \cdot t_3$	$t_3 \leftarrow 3X^2 + a \cdot Z_2^2$
$\mathbf{X} \leftarrow t_2^2 - (t_4 + 2t_5)$	$\mathbf{X} \leftarrow t_3^2 - 2t_2$
$\mathbf{Y} \leftarrow t_2 \cdot (t_5 - \mathbf{X}) - Y \cdot t_4$	$\mathbf{Y} \leftarrow t_3 \cdot (t_2 - \mathbf{X}) - 8t_1^2$
$\mathbf{Z} \leftarrow Z \cdot t_1$	$\mathbf{Z} \leftarrow Z \cdot 2Y$
$\lambda_d \leftarrow \mathbf{Z}$	$\lambda_d \leftarrow \mathbf{Z} \cdot Z_2$
$\lambda_n \leftarrow \lambda_d \cdot (y_P - y_Q) - t_2 \cdot (x_P - x_Q)$	$\lambda_n \leftarrow \lambda_d \cdot y_P - 2t_1 - t_3 \cdot (Z_2 \cdot x_P - X)$
return $((\lambda_n, \lambda_d), \mathbf{S} = (\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{Z}^2))$	**return** $((\lambda_n, \lambda_d), \mathbf{S} = (\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{Z}^2))$

Algorithm 5: VerticalLine($S \in E(\mathbb{F}_{p^k}), P \in E(\mathbb{F}_p)$) – Compute the line through S and $-S$ evaluated at P.

$(X, Y, Z, Z_2) \leftarrow S$; $(x_P, y_P) \leftarrow P$
return $(\mu_n = Z_2, \mu_d = Z_2 \cdot x_P - X)$

k	$-D$	AddLine and DoubleLine	Vertical Line	Update1 and Update2	ref
Weierstrass model					
5	any	$10\mathbf{m}_5 + 3\mathbf{s}_5$ $6\mathbf{m}_5 + 4\mathbf{s}_5 + 10\mathbf{m}$	$5\mathbf{m}$	$4\mathbf{m}_5 + 2\mathbf{s}_5$ $4\mathbf{m}_5$	Alg. 4,5
7	any	$10\mathbf{m}_7 + 3\mathbf{s}_7$ $6\mathbf{m}_7 + 4\mathbf{s}_7 + 14\mathbf{m}$	$7\mathbf{m}$	$4\mathbf{m}_7 + 2\mathbf{s}_7$ $4\mathbf{m}_7$	Alg. 4,5
$6 \mid k$	-3	$10\mathbf{m}_{k/6} + 2\mathbf{s}_{k/6} + (k/3)\mathbf{m}$ $2\mathbf{m}_{k/6} + 7\mathbf{s}_{k/6} + (k/3)\mathbf{m}$	0	$\mathbf{s}_k + 13\mathbf{m}_{k/6}$ $13\mathbf{m}_{k/6}$	[21, §5]
$6 \mid k$	-3	$11\mathbf{m}_{k/6} + 2\mathbf{s}_{k/6} + (k/3)\mathbf{m}$ $3\mathbf{m}_{k/6} + 6\mathbf{s}_{k/6} + (k/3)\mathbf{m}$	0	$\mathbf{s}_k + 13\mathbf{m}_{k/6}$ $13\mathbf{m}_{k/6}$	[5, §4,6]
$4 \mid k$	-4	$9\mathbf{m}_{k/4} + 5\mathbf{s}_{k/4} + (k/2)\mathbf{m}$ $2\mathbf{m}_{k/4} + 8\mathbf{s}_{k/4} + (k/2)\mathbf{m}$	0	$\mathbf{s}_k + 8\mathbf{m}_{k/4}$ $8\mathbf{m}_{k/4}$	[21, §4]
Jacobi quartic model (not for cubic or sextic twist)					
$4 \mid k$	-4	$12\mathbf{m}_{k/4} + 7\mathbf{s}_{k/4} + 1\mathbf{mc}_{k/4} + (k/2)\mathbf{m}$ $3\mathbf{m}_{k/4} + 7\mathbf{s}_{k/4} + 1\mathbf{mc}_{k/4} + (k/2)\mathbf{m}$	0	$\mathbf{s}_k + 8\mathbf{m}_{k/4}$ $8\mathbf{m}_{k/4}$	[23]
$2 \mid k$	any	$16\mathbf{m}_{k/2} + 1\mathbf{s}_{k/2} + 4\mathbf{mc}_{k/2} + k\mathbf{m}$ $4\mathbf{m}_{k/2} + 8\mathbf{s}_{k/2} + 1\mathbf{mc}_{k/2} + k\mathbf{m}$	0	$\mathbf{s}_k + \mathbf{m}_k$ $\mathbf{m}_k$	[25, §3.2]
Hessian model (not for quartic twist)					
$6 \mid k$	-3	$7\mathbf{m}_{k/3} + 4\mathbf{m}_{k/6} + (2k/3)\mathbf{m}$ $2\mathbf{m}_{k/6} + \mathbf{s}_{k/6} + 4\mathbf{m}_{k/3} + 2\mathbf{s}_{k/3} + (k/2)\mathbf{m}$	0	$\mathbf{s}_k + 13\mathbf{m}_{k/6}$ $2\mathbf{m}_{k/3}$	[18]
Edwards model (not for quartic, cubic or sextic twist)					
$2 \mid k$	any	$12\mathbf{m}_{k/2} + \mathbf{mc}_{k/2} + k\mathbf{m}$ $4\mathbf{m}_{k/2} + 7\mathbf{s}_{k/2} + 2\mathbf{mc}_{k/2} + k\mathbf{m}$	0	$\mathbf{s}_k + \mathbf{m}_k$ $\mathbf{m}_k$	[40,25]

Table 6: Miller loop cost (see Equation (1)). We assume $a = 0$ when $6 \mid k$ and $D = 3$, $b = 0$ when $4 \mid k$ and $D = 4$, and $a = -3$ otherwise. The second option for $6 \mid k$ is reported by [5, §4] to perform slightly better.

k	$(p^k - 1)/\Phi_k(p)$	$\mathbf{c}_{\textsc{FirstExp}}$	comment
5	$p - 1$	$(4\mathbf{m}) + \mathbf{i}_5 + \mathbf{m}_5$	Can omit $\mathbf{f}_5 = 4\mathbf{m}$ which appears in $\mathbf{i}_5$
6	$(p+1)(p^3-1)$	$\mathbf{f}_6 + \mathbf{m}_6 + 2\mathbf{s}_3 + 3\mathbf{m}_3 + \mathbf{i}_3$ $= 4\mathbf{m} + \mathbf{m}_6 + \mathbf{i}_6 + \mathbf{m}_3$	Uses $\mathbb{F}_{p^6} = \mathbb{F}_{p^3}(y) = \mathbb{F}(x)(y)$ with $y^2 = x$ and $x^3 = \beta$
7	$p - 1$	$(6\mathbf{m}) + \mathbf{i}_7 + \mathbf{m}_7$	Can omit $\mathbf{f}_7 = 6\mathbf{m}$ which appears in $\mathbf{i}_7$
8	$p^4 - 1$	$\mathbf{i}_8 + \mathbf{m}_8$	

Table 7: Cost $\mathbf{c}_{\textsc{FirstExp}}$ of the first part of the final exponentiation

The first part of the exponentiation uses few Frobenius powers and inversions and its cost (Table 7) depends on the value of $\Phi_k(p)$. Its computation is very efficient because of Frobenius powers (Remark 2). In particular, for $x \in \mathbb{F}_{p^8}$, x^{p^4} is almost free: it is simply the conjugate of x seen in a quadratic extension of $\mathbb{F}_{p^4}$.

The second part of the exponentiation is more expensive and is specific to each curve. The key ingredient is the base-p representation of the exponent, since Frobenius powers p^i are computed efficiently. Notice that in Algorithm 2, we have $p \equiv (t-1) \equiv (t_0 - 1) \mod r$. Let c be such that $p + 1 - t_0 = c \cdot r$. The expression $(\Phi_k(p) - \Phi_k(t_0-1))/r$ simplifies, and we obtain a nice generic formula in p and t_0 for each embedding degree. The actual expression depends on the exponent i in Algorithm 2, as well as on congruence conditions on T. We only detail a few examples. Formulas for the other cases can be obtained with the companion software mentioned in §1.1.

For $k = 8$, we choose $D = 4$ so that $\sqrt{-D} = 2T^2$. When for instance we choose $i = 5$ in Algorithm 2, we have $t_0 = T^5 + 1 \bmod r = 1 - T$. This leads to the following expression, where h_u denotes the integer $(h_t + 1)/2$.

$$\begin{aligned}\Phi_8(p)/r &= \Phi_8(t_0 - 1)/r + (p + t_0 - 1)(p^2 + (t_0 - 1)^2))c, \\ c &= ((h_u^2 - h_u + h_y^2 + 1/4)T - h_y)T - h_y + 1/4)T \\ &\quad -h_u + 1)T + h_u^2 + h_y^2 \end{aligned} \tag{2}$$

where $\Phi_8(t_0 - 1)/r = \Phi_8(-T)/r = 1$ by construction. To raise to the power $\Phi_8(p)/r$, we use the fact that T is even to deal with fractional values in the exponent. We obtain the following upper bound on the cost, using $\mathbf{c}_T$, $\mathbf{c}_u$, $\mathbf{c}_y$ to denote the cost of raising to the power T, h_u, and h_y, respectively:

$$\mathbf{c}_{\text{SecondExp},k=8} = (3\mathbf{f}_k + 2\mathbf{s}_k + 3\mathbf{m}_k) + (11\mathbf{m}_k + 4\mathbf{c}_T + 2\mathbf{c}_u + 2\mathbf{c}_y).$$

For $k = 6$, we choose $D = 3$ so that $\sqrt{-D} = 2T - 1$. We obtain expressions that vary slightly depending on i, and on the congruence class of $h_t \bmod 2$ and $T \bmod 3$. It also appears that it is more convenient to compute the cube of the pairing. When for instance we choose $i = 1$ in Algorithm 2, and that $T \bmod 3 = 1$ and $h_t \bmod 2 = 1$, we have the following expression, where $u = (h_t + 1)/2$, $w = h_y/2$, and $T' = T - (T \bmod 3) = T - 1$:

$$\begin{aligned}3\Phi_6(p)/r &= 3\Phi_6(t_0 - 1)/r + 3(p + t_0)c, \\ 3c &= ((3u^2 + 9w^2 - 3u - 3w + 1)T' + \\ &\quad 3u^2 + 9w^2 - 6w)T' + 3u^2 + 9w^2 + 3u - 9w. \end{aligned} \tag{3}$$

Raising to the power $3\Phi_k(p)/r$ thus has the following cost (we give an upper bound on all possible congruence conditions). We use $\mathbf{c}_u$, $\mathbf{c}_w$, $\mathbf{c}_T$ and $\mathbf{c}_{T'}$ to denote the cost of raising to the powers u, w, T and $T' = T - (T \bmod 3)$, respectively.

$$\mathbf{c}_{\text{SecondExp},k=6} = (\mathbf{c}_T + \mathbf{f}_k + 2\mathbf{s}_k + 4\mathbf{m}_k) + (12\mathbf{m}_k + 2\mathbf{s}_k + 2\mathbf{c}_u + 2\mathbf{c}_w + 2\mathbf{c}_{T'})$$

For $k=5$ and $k=7$, we use $p=(t_0-1)+c\Phi_k(T)$ to reduce $\Phi_k(p)/\Phi_k(T)$ (rational fraction in the indeterminate T) to the form

$$\Phi_k(p)/r = \sum_{0\le j\le k-2} p^j a_j(c,T).$$

The exact expression of the coefficients $(a_j)_j$ depends on k and i, and so does the cost of raising to these powers. For example, for $k=5$ and $i=2$, we have

$$(a_j)_{0\le j\le 3} = (-cT^3-T+1, -cT^3-(c+1)T+1, cT^2+c+1, c).$$

By applying this method, we found that for $k=5$ and $k=7$, raising to the power $\Phi_k(p)/r$ costs at most

$$\mathbf{c}_{\textsc{SecondExp},k\in\{5,7\}} = 2\mathbf{i}_k + (k-2)(\mathbf{f}_k + \mathbf{c}_T + 2\mathbf{m}_k) + \mathbf{c}_c + \mathbf{m}_k$$

where the two inversions can be saved in the favorable case $i=1$, and $\mathbf{c}_T$ and $\mathbf{c}_c$ are the costs of raising to the powers T and c, respectively.

6 Comparisons with previous curves

We compare curves generated with Algorithm 2 of embedding degree 5 to 8 with the state of the art: BN and BLS12 curves [3], KSS16 curves [21, §4], and $k=1$ curves [16]. Note that several estimates below differ marginally from [7], which uses a different estimated cost $\mathbf{i}_{12}=\mathbf{i}_1+97\mathbf{m}$, and also reproduce the $7\mathbf{m}_2$ estimate that we mentioned in footnote 3 on page 12.

BN curve with a 462-bit prime p. Barbulescu and Duquesne give in [7] parameters of a BN curve for 128 bits of security. The curve is defined from the parameter $u=2^{114}+2^{101}-2^{14}-1$ and has a prime p of 462 bits (see Table 2). An estimation of the optimal ate pairing on this curve is also given in [7], we reproduce the final count. The Miller loop iterates on $T=6u+2$, of 117 bits and NAF-Hamming-weight 7. Reusing Equation (1) and Tables 5 and 6, and taking into account the correcting terms of the Miller loop for BN curves [50], we get:

$$\begin{aligned}\mathbf{c}_{\textsc{MillerLoop}} &= 116(3\mathbf{m}_2+6\mathbf{s}_2+4\mathbf{m})+115(\mathbf{s}_{12}+13\mathbf{m}_2)\\ &\quad +6(11\mathbf{m}_2+2\mathbf{s}_2+4\mathbf{m}+13\mathbf{m}_2)\\ &\quad +(11\mathbf{m}_2+2\mathbf{s}_2+4\mathbf{m})+4\mathbf{m}_2+4\mathbf{m}+2(13\mathbf{m}_2)+4(10\mathbf{m})\\ &= 12180\mathbf{m}.\end{aligned}$$

According to [33, Corollary 4.1], raising to the power u costs

$$\mathbf{c}_u = 4(114-1)\mathbf{m}_2 + (6\cdot 3-3)\mathbf{m}_2 + 3\mathbf{m}_{12} + 3\cdot 3\mathbf{s}_2 + \mathbf{i}_2 = 1585\mathbf{m}+\mathbf{i}.$$

The final exponentiation costs

$$\mathbf{c}_{\textsc{FinalExp}} = \mathbf{i}_{12} + 12\mathbf{m}_{12} + 3\mathbf{s}_{12}^{\text{cyclo}} + 4\mathbf{f}_{12} + 3\mathbf{c}_u = 5591\mathbf{m}+4\mathbf{i}$$

where $\mathbf{s}_{12}^{\text{cyclo}}$ is the cost of cyclotomic squarings (see [33]), namely $\mathbf{s}_{12}^{\text{cyclo}}=18\mathbf{m}$. The optimal ate pairing on the BN-462 curve costs in total $17771\mathbf{m}+4\mathbf{i}$.

BLS12 curve with a 461-bit prime p. We reproduce the results from [3] adapted to the parameter $u = -2^{77} + 2^{50} + 2^{33}$ from [7]. We obtain:

$$\begin{aligned}\mathbf{c}_{\textsc{MillerLoop}} &= 76(3\mathbf{m}_2 + 6\mathbf{s}_2 + 4\mathbf{m}) + 75(\mathbf{s}_{12} + 13\mathbf{m}_2)\\ &\quad +2(11\mathbf{m}_2 + 2\mathbf{s}_2 + 4\mathbf{m} + 13\mathbf{m}_2)\\ &= 7685\mathbf{m}.\end{aligned}$$

As above, we adapt [33, Corollary 4.1]. Raising to the power u costs

$$\begin{aligned}\mathbf{c}_u &= 4(77-1)\mathbf{m}_2 + (6\cdot 1 - 3)\mathbf{m}_2 + 2\mathbf{m}_{12} + 3\cdot 2\cdot \mathbf{s}_2 + \mathbf{i}_2\\ &= 1045\mathbf{m} + \mathbf{i}.\end{aligned}$$

The final exponentiation costs

$$\begin{aligned}\mathbf{c}_{\textsc{FinalExp}} &= \mathbf{i}_{12} + 12\mathbf{m}_{12} + 2\mathbf{s}_{12}^{\text{cyclo}} + 4\mathbf{f}_{12} + 5\mathbf{c}_u\\ &= 6043\mathbf{m} + 6\mathbf{i}.\end{aligned}$$

The optimal ate pairing on the BLS12-461 curve costs in total $13728\mathbf{m} + 6\mathbf{i}$.

KSS16 curve with a 339-bit prime p. We reproduce the results from [21] with the parameter $u = 2^{35} - 2^{32} - 2^{18} + 2^8 + 1$ from [7]. We obtain:

$$\begin{aligned}\mathbf{c}_{\textsc{MillerLoop}} &= 34(2\mathbf{m}_4 + 8\mathbf{s}_4 + 8\mathbf{m}) + 33(\mathbf{s}_{16} + (8\mathbf{m}_4)) + 4(9\mathbf{m}_4 + 5\mathbf{s}_4 + 8\mathbf{m})\\ &\quad + 3(14\mathbf{m}) + 5\mathbf{m}_4 + \mathbf{s}_4 + 16\mathbf{m} + 6(8\mathbf{m}_4)\\ &= 7691\mathbf{m}.\end{aligned}$$

Raising to the power u costs $34\mathbf{s}_{16}^{\text{cyclo}} + 4\mathbf{m}_{16} = 1548\mathbf{m}$, and the final exponentiation costs:

$$\begin{aligned}\mathbf{c}_{\textsc{FinalExp}} &= \mathbf{i}_{16} + 32\mathbf{m}_{16} + 34\mathbf{s}_{16}^{\text{cyclo}} + 8\mathbf{f}_{16} + 24\mathbf{m}_4 + 9(1684\mathbf{m})\\ &= 18210\mathbf{m} + \mathbf{i}.\end{aligned}$$

The optimal ate pairing on the KSS16-339 curve costs in total $25901\mathbf{m} + \mathbf{i}$.

Curves of embedding degree one. The curves suggested by [16] are resistant to TNFS because the target finite field is $\mathbb{F}_p$, with p as large as 3072 bits. The ate pairing is not available on these curves because the trace is $t = 2$, so the Tate pairing must be used. Its cost is given in [16]: for a 256-bit r, the Miller loop costs $4626\mathbf{m} + \mathbf{i}$ and the final exponentiation costs $4100\mathbf{m}$. The total cost is finally $8726\mathbf{m} + \mathbf{i}$.

6.1 Our new STNFS-resistant curves at the 128-bit security level

We generate four curves of embedding degree 5, 6, 7 and 8 with Algorithm 2 combined with the CM method and estimate the cost of the new optimal ate pairing on these curves. Code to reproduce this search can be found in the repository mentioned in §1.1.

Twist-secure and subgroup-secure parameters. We checked our curves for twist- and subgroup-security (see [10]). For each curve, we checked the size of the cofactors of the curve and its quadratic twist on $\mathbb{G}_1$ and $\mathbb{G}_2$. A curve E is η-subgroup-secure over $\mathbb{F}_q$ if all the factors of $E(\mathbb{F}_q)$ are at least as large as r, except those of size η. A curve is twist-subgroup secure if its quadratic twist is subgroup-secure. This makes five criteria: subgroup- and twist-subgroup- security for both $\mathbb{G}_1$ and $\mathbb{G}_2$, as well as subgroup security for $\mathbb{G}_T$ (with respect to $\Phi_k(p)$). We selected four curves that are subgroup and twist-subgroup secure for $\mathbb{G}_1$ with $\eta = 10$. Except in the $k = 7$ case, the curve containing the subgroup $\mathbb{G}_2$ is also twist-subgroup secure. We did not investigate the $\mathbb{G}_T$ subgroup-security: together with the Cocks–Pinch conditions, it would require finding parameters such that $\Phi_k(p)/r$ is prime or almost prime. With the sizes provided in the following paragraph, $1088 \leq \log_2(\Phi_k(p)/r) \leq 2816$ so it is impossible to factor this thousand-bit integer entirely and it will very unlikely be prime.

Parameter choices. We explain here the choices for choosing parameter sizes in Algorithm 2:

Size of the prime p We target a size for the finite field $\mathbb{F}_{p^k}$ that determines the size of p. These values can be read in Table 3.

Hamming weight (or 2-NAF weight) of T We restrict to low weight T in order to get an efficient Miller loop. We choose $\mathrm{HW}_{2-\mathrm{NAF}}(T) = 4$ for the $k = 5$ curve, or $\mathrm{HW}_{2-\mathrm{NAF}}(T) = 5$ for others.

Discriminant For efficiency, we target curves with as many automorphisms as possible. For $k = 6$ (resp. 8), we set $D = 3$ (resp. 4) so that a sextic (resp. quartic) twist is avaible. For $k = 5$, we chose arbitrarily $D \approx 10^{10}$, which is well within the feasible range for the CM method. For $k = 7$, the size of p (512 bits) restricts us to small discriminants, since we must have $4p = t^2 + Dy^2$ with $\log_2(t), \log_2(y) \approx \log_2(r) = 256$.

Hamming weight (or 2-NAF weight) of h_t and h_y As explained in §5.2, for $k = 6$ and $k = 8$ we restrict to low weight cofactors h_t and h_y so as to accelerate the exponentiation to the power c in the second part of the final exponentiation (see Equations (2) and (3)).

Allowing a cofactor of 10 bits, we obtain twist- and subgroup-secure elliptic curves of embedding degree five to eight. We denote by E^t the quadratic twist of E and $\tilde{E}$ the degree d twist of E such that $E(\mathbb{F}_{p^k})[r] \simeq \tilde{E}(\mathbb{F}_{p^{k/d}})[r]$. Points on E can be represented with the Edwards model if we restrict to curves with 4-torsion points. For the remainder of this section, the notation p_N denotes an arbitrary prime of N bits.

Curve of embedding degree 5 The curve $E : y^2 = x^3 - 3x + b_5$ defined over $\mathbb{F}_p$ with

$$\begin{aligned} b_5 = {} & \texttt{0x3dd2d2b0b2e68770bf01b41946ab867390cf9ecc4a858004fc769c} \\ & \texttt{278f079574677c7db3e7201c938b099f85eb6e85f200b95a80b24fdb} \\ & \texttt{df584098d690c6b91b21d00f52cc79473a11123b08ab2a616b4a4fbf} \\ p = {} & \texttt{0x40000138cd26ab94b86e1b2f7482785fa18f877591d2a4476b4760} \\ & \texttt{217f860bfe8674e2a4610d669328bda13044c030e8cc836a5b363f2d} \\ & \texttt{4c8abcab71b12091356bb4695c5626bc319d38bf65768c5695f9ad97} \end{aligned}$$

satisfies

$$\#E(\mathbb{F}_p) = 2^2 \cdot p_{405} \cdot r \quad \#\tilde{E}(\mathbb{F}_{p^5}) = p_{2393} \cdot (2^2 \cdot p_{405} \cdot r) \cdot r$$
$$\#E^t(\mathbb{F}_p) = 2^2 \cdot p_{661} \quad \#E^t(\mathbb{F}_{p^5}) = p_{2649} \cdot (2^2 \cdot p_{661})$$

$$r = \texttt{0x9610000000015700ab80000126012600c4007000a800e000f000200040008001}$$

The additional parameters to obtain the curve from Algorithm 2 are :

$$T = 2^{64} - 2^{61} + 2^{15}, D = 10^{10} + 147, i = 1, h_t = 3, h_y = -\texttt{0x11e36418c7c8b454}$$

and $\mathbb{F}_{p^5}$ can be defined as $\mathbb{F}_p[x]/(x^5 - 5)$.

Curve of embedding degree 6 The curve $E : y^2 = x^3 - 1$ defined over $\mathbb{F}_p$ with

$$\begin{aligned} p = {} & \texttt{0x9401ff90f28bffb0c610fb10bf9e0fefd59211629a7991563c5e468} \\ & \texttt{d43ec9cfe1549fd59c20ab5b9a7cda7f27a0067b8303eeb4b31555cf4} \\ & \texttt{f24050ed155555cd7fa7a5f8aaaaaaad47ede1a6aaaaaaaab69e6dcb} \end{aligned}$$

satisfies

$$\#E(\mathbb{F}_p) = 2^2 \cdot p_{414} \cdot r \quad \#\tilde{E}(\mathbb{F}_p) = 3 \cdot p_{414} \cdot r$$
$$\#E^t(\mathbb{F}_p) = 2^2 \cdot 3 \cdot 7 \cdot p_{665} \quad \#E^t(\mathbb{F}_p) = 13 \cdot 19 \cdot p_{664}$$

$$r = \texttt{0xe0ffffffffffffc400000000000003ff1000000000000020000000000000001}$$

The additional parameters to obtain the curve from Algorithm 2 are :

$$T = 2^{128} - 2^{124} - 2^{69}, D = 3, i = 1, h_t = -1, h_y = 2^{80} - 2^{70} - 2^{66} - \texttt{0x3fe0}$$

and $\mathbb{F}_{p^6}$ can be defined as $\mathbb{F}_p[x]/(x^6 - 2)$.

Curve of embedding degree 7 The curve $E : y^2 = x^3 - 3u^2x + b_7u^3$ with u non-square defined over $\mathbb{F}_p$ with

$$\begin{aligned} b_7 = {} & \texttt{0x15d384c76889d377dd63600fbe42628e0c386a3e87} \\ & \texttt{915790188d944845aab2b649964f386dc90b3a9b612} \\ & \texttt{0af5da9a2aaead5e415dd958c5cfa80ea61aac268b0} \\ p = {} & \texttt{0x8f591a9876a6d2344ae66dd7540ea2fd28174755d1} \\ & \texttt{6c4ae5c5cd5c1d208e639271b48c8ba7453c95a2a9b} \\ & \texttt{e6434f2455504d419f13e35062aa5ebbc49ecfd30f9} \end{aligned}$$

satisfies

$$\#E(\mathbb{F}_p) = 2^2 \cdot 3^2 \cdot p_{251} \cdot r \quad \#E^t(\mathbb{F}_{p^7}) = 2^5 \cdot 5 \cdot p_{504}$$

$$r = \texttt{0xb63ccd541c3aa13c7b7098feb312eecf5648fd215c0d2916714b429d14e8f889}$$

The additional parameters to obtain the curve from Algorithm 2 are :

$$T = 2^{43} - 2^{41} - \texttt{0x47dfdb8}, D = 20, i = 6, h_t = -2, h_y = 0$$

and $\mathbb{F}_{p^7}$ can be defined as $\mathbb{F}_p[x]/(x^7 - 2)$.

Curve of embedding degree 8 The curve $E : y^2 = x^3 + 289x$ defined over $\mathbb{F}_p$ with

$$\begin{aligned} p = {} & \texttt{0xb4910005e588fde4023747293d3a6e3d41b42afe599c} \\ & \texttt{f6ed3e0192d99fe38524365563d4dd1749878641cde159} \\ & \texttt{afdb73b758c3baa70c8c1fa842a7142d6a5981846aba09} \end{aligned}$$

satisfies

$$\#E(\mathbb{F}_p) = 2^2 \cdot p_{286} \cdot r \quad \#\tilde{E}(\mathbb{F}_{p^2}) = 2 \cdot p_{830} \cdot r$$

$$\#E^t(\mathbb{F}_p) = 2^4 \cdot 29 \cdot p_{535} \quad \#E^t(\mathbb{F}_{p^2}) = 2 \cdot p_{1086}$$

$$r = \texttt{0xffffffffbfdf08000606301799d79ce503fe520538262507b940781000000001}$$

The additional parameters to obtain the curve from Algorithm 2 are :

$$T = 2^{64} - \texttt{0x10083e00}, D = 4, i = 5, h_t = 5, h_y = -\texttt{0xd700}$$

and $\mathbb{F}_{p^8}$ can be defined as $\mathbb{F}_p[x]/(x^8 - 17)$.

6.2 Comparison with the state of the art

In order to compare the pairing on our curves with the computation on BN, BLS12, KSS16, and $k = 1$ curves, we need to determine the cost of a multiplication $\mathbf{m}$ for different sizes of p. Indeed, a multiplication in the 3072-bit field of $k = 1$ curves is much more expensive than in a 512-bit prime field. Table 8 shows the benchmarks with RELIC [4] for base field arithmetic with the different primes involved in our pairings.

Pairing computation. The costs of the Miller loop and the first part of the final exponentiation are given by Equation (1), Table 6, and Table 7. The second part of the final exponentiation is covered by §5.2. This part is specific to each set of curve parameters, in particular $\mathrm{HW}_{\text{2-NAF}}(h_t)$ and $\mathrm{HW}_{\text{2-NAF}}(h_y)$. Appendix A goes into more detail for our curve with $k = 8$. Further detail for the second part of the final exponentiation for all exponents, covering the various cases, can be found in the code repository (see §1.1).

Table 9 summarises our comparison results for pairing computations. We warn the reader that timings of Table 9 are not real pairing computations, as we simply used as a base the arithmetic operations of RELIC [4], and the multiplication costs that we detailed in the paragraphs above. This being said, for the curves where an actual implementation of the optimal ate pairing is available with RELIC (BN and BLS12 curves), the estimation that we obtain is within 10% of the actual computation time. This gives reasonable confidence for the validity of the other projected timings.

Miller loop. We obtain a faster Miller loop for $k = 6$ and $k = 8$ curves compared to BN and BLS12 curves. The $k = 8$ curve has a shorter Miller loop (64-bit) compared to the BN and BLS12 ones (117-bit). The $k = 6$ curve has a sextic twist that allows to compute $f_{T,Q}(P)$ on $\mathbb{F}_p$ of 672 bits, compared to a field of 922 bits for BN and BLS12 curves. As for the cases $k = 5$ and $k = 7$, the Miller loop is not as efficient because no twist is available, and the computation is done over $\mathbb{F}_{p^k}$. Comparisons between $k = 6$ and $k = 7$ curves show that using a curve with twists is a better option than having a short Miller loop. The best option is obviously to have a short Miller loop *and* a curve with twists, as for $k = 8$ curves.

Final exponentiation. The rewriting tricks used in §5.2 for the final exponentiation apply for any curve obtained with Algorithm 2 with the optimisation $r = \Phi_k(T)$. For $k = 6$ and $k = 8$ the cofactor is smaller, and the discriminant $D = 3$, resp. $D = 4$, gives formulas that are as good as for BN and BLS12 curves. For $k = 5$ and $k = 7$ curves, the exponentiation is less efficient because fast cyclotomic squaring formulas are not available.

Total cost. Table 9 shows that our new pairing is almost as efficient as the optimal ate pairing on the BLS12 and KSS16 curves. Given the nature of Table 9 which gives *estimated* timings, it is however more appropriate to say that the performance difference is within the error margin. Additionally, we estimate that the optimal ate pairing on our $k = 8$ curve is up to 23 times more efficient than the Tate pairing on $k = 1$ curves [16].

Prime size	**Building block for**	**$\mathbb{F}_p$ multiplication**
$192 < \log_2(p) \leq 256$		35ns
$320 < \log_2(p) \leq 384$	KSS16	69ns
$448 < \log_2(p) \leq 512$	BN, BLS12, $k = 7$	120ns
$512 < \log_2(p) \leq 576$	$k = 8$	154ns*
$576 < \log_2(p) \leq 640$		188ns
$640 < \log_2(p) \leq 704$	$k = 5$, $k = 6$	230ns*
$3008 < \log_2(p) \leq 3072$	[16]	4882ns**

Table 8: $\mathbb{F}_p$ multiplication timing for RELIC on a Intel Core i5-4570 CPU, 3.20GHz

*Estimation because no bench is available for 9 and 11 machine words primes.

**Benched with GNU MP

Curve	**Prime**	**Miller loop** time estimation	**Exponentiation** time estimation	**Total**	time estimation
$k = 5$	663-bit	14500**m** 3.3ms	9813**m** 2.3ms	24313**m**	5.6ms
$k = 6$	672-bit	4601**m** 1.1ms	3871**m** 0.9ms	8472**m**	2.0ms
$k = 7$	512-bit	18342**m** 2.2ms	13451**m** 1.6ms	31793**m**	3.8ms
$k = 8$	544-bit	4502**m** 0.7ms	7134**m** 1.1ms	11636**m**	1.8ms
BN	462-bit	12180**m** 1.5ms	5691**m** 0.7ms	17871**m**	2.2ms
BLS12	461-bit	7685**m** 0.9ms	6193**m** 0.7ms	13878**m**	1.6ms
KSS16	339-bit	7691**m** 0.5ms	18235**m** 1.3ms	25926**m**	1.8ms
$k = 1$	3072-bit	4651**m** 22.7ms	4100**m** 20.0ms	8751**m**	42.7ms

Table 9: Pairing cost and timing extrapolation from Table 8

Elliptic curve scalar multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$. Our generation of curve leads to large prime value (up to eleven 64-bit words instead of eight for BN and BLS12 curves). The scalar multiplication cost on $\mathbb{G}_1$ is not affected by slow finite field multiplications because our curves benefit of other improvements: BN (resp. BLS) curves paramters for 128 bits of security lead to scalar multiplications $[k]P$ on $\mathbb{G}_1$ and $\mathbb{G}_2$ with $\log_2(k) \approx 448$ (resp. 300). For our curves of embedding degree five to eight, we choose r of minimal size (256 bits to withstand the Pollard rho attack). Some curves get benefits of efficient group law arithmetic: curves of embedding degree 5, 7, and 8 use the Edwards model. The $k = 6$ curve use the efficient formulas available for $a = 0$ curves, widely used in practice. The Gallant–Lambert–Vanstone (GLV) method can be performed on $k = 6$ and $k = 8$ curves in order to reduce the number of doubling and addition steps. Over $\mathbb{G}_2$, the scalar multiplication is often accelerated by using a twist of the curve. The trick is available for curves of degree 6 and 8, but not for $k = 5$ and 7. Even if the main topic of this paper is about pairing computations, various protocols also compute scalar multiplications. Curves of embedding degree 5 and 7 do not benefit of twists and GLV optimisation, so the cost over $\mathbb{G}_2$ is too expensive for practical applications.

7 Conclusion

We modified the Cocks–Pinch method to generate pairing-friendly elliptic curves with an optimal Miller loop length $\log r/\varphi(k)$ to compute efficiently an optimal ate pairing. Moreover the parameters are carefully chosen so that the curves withstand the recent STNFS attacks on discrete logarithms. Optimal ate pairing computation on our $k = 8$ curve seems to be faster than on the BN curves proposed in [7]. In the $k = 8$ case, performance is apparently on par with BLS12 curves. Compared to $k = 1$ curves presented in [16], pairing computations on the curves suggested here are expected to be 7 ($k = 5$) to 23 ($k = 8$) times faster.

One lesson of our work is that the Miller loop length is very important for an efficient pairing, even in the Cocks–Pinch case. It matters much more than the ρ value.

With respect to the threats on pairing summarised in Table 1, users fearing the progress of NFS variants should prefer the more conservative choice of our modified Cocks–Pinch curves: unlike BN, BLS12, and KSS16 curves, we do not have to use much larger parameters to be STNFS-resistant.

We finally note that the short Miller loop on our $k = 6$ and $k = 8$ curves is well-suited to protocols where the product of several pairings is computed, the final exponentiation being computed only once, after the Miller loops. This is the case for the translation in the prime-order setting of the Boneh–Boyen IBE scheme: the product of six pairings is computed in the decryption step, and for the hierarchical identity-based encryption based on Lewko–Waters scheme: the product of ten pairings is computed in the decryption step [39].

Acknowledgements. The second author thanks P. Zimmermann for his help with Pari/Gp, and P. Gaudry and T. Kleinjung for their contribution to Table 11.

References

1. Aoki, K., Franke, J., Kleinjung, T., Lenstra, A.K., Osvik, D.A.: A kilobit special number field sieve factorization. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 1–12. Springer, Heidelberg (Dec 2007). https://doi.org/10.1007/978-3-540-76900-2_1
2. Aranha, D.F.: Pairings are not dead, just resting (nov 2017), slides at ECC 2017 workshop. https://ecc2017.cs.ru.nl/
3. Aranha, D.F., Fuentes-Castañeda, L., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Implementing pairings at the 192-bit security level. In: Abdalla, M., Lange, T. (eds.) PAIRING 2012. LNCS, vol. 7708, pp. 177–195. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-36334-4_11
4. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic
5. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_5
6. Bai, S., Gaudry, P., Kruppa, A., Thomé, E., Zimmermann, P.: Factorization of RSA-220. Number Theory list (May 12 2016), https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;d17fe291.1605, http://www.loria.fr/~zimmerma/papers/rsa220.pdf
7. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Journal of Cryptology (Jan 2018). https://doi.org/10.1007/s00145-018-9280-5

8. Barbulescu, R., Gaudry, P., Guillevic, A., Morain, F.: Improving NFS for the discrete logarithm problem in non-prime finite fields. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 129–155. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_6
9. Barbulescu, R., Gaudry, P., Kleinjung, T.: The tower number field sieve. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 31–55. Springer, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48800-3_2
10. Barreto, P.S.L.M., Costello, C., Misoczki, R., Naehrig, M., Pereira, G.C.C.F., Zanon, G.: Subgroup security in pairing-based cryptography. In: Lauter, K.E., Rodríguez-Henríquez, F. (eds.) LATINCRYPT 2015. LNCS, vol. 9230, pp. 245–265. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-319-22174-8_14
11. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) ITCS 2012. pp. 326–349. ACM (Jan 2012). https://doi.org/10.1145/2090236.2090263
12. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_13
13. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_30
14. Bouvier, C., Gaudry, P., Imbert, L., Jeljeli, H., Thomé, E.: Discrete logarithms in GF(p) — 180 digits. Number Theory list, item 004703 (June 11 2014), https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;615d922a.1406
15. Bowe, S.: BLS12-381: New zk-SNARK elliptic curve construction. Zcash blog (March 11 2017), https://blog.z.cash/new-snark-curve/
16. Chatterjee, S., Menezes, A., Rodríguez-Henríquez, F.: On instantiating pairing-based protocols with elliptic curves of embedding degree one. IEEE Trans. Computers **66**(6), 1061–1070 (2017). https://doi.org/10.1109/TC.2016.2633340
17. Childers, G.: Factorization of a 1061-bit number by the special number field sieve. Cryptology ePrint Archive, Report 2012/444 (2012), http://eprint.iacr.org/2012/444
18. Chuengsatiansup, C., Martindale, C.: Pairing-friendly twisted hessian curves. In: Chakraborty, D., Iwata, T. (eds.) INDOCRYPT. LNCS, vol. 11356, pp. 228–247. Springer, New Delhi, India (December 9-12 2018). https://doi.org/10.1007/978-3-030-05378-9_13
19. Chung, J., Hasan, M.A.: Asymmetric squaring formulae. In: 18th IEEE Symposium on Computer Arithmetic (ARITH '07). pp. 113–122 (June 2007). https://doi.org/10.1109/ARITH.2007.11
20. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT'98. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (Oct 1998). https://doi.org/10.1007/3-540-49649-1_6
21. Costello, C., Lange, T., Naehrig, M.: Faster pairing computations on curves with high-degree twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-13013-7_14
22. Devegili, A.J., Ó hÉigeartaigh, C., Scott, M., Dahab, R.: Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471 (2006), http://eprint.iacr.org/2006/471

23. Duquesne, S., El Mrabet, N., Fouotsa, E.: Efficient computation of pairings on jacobi quartic elliptic curves. J. Mathematical Cryptology **8**(4), 331–362 (2014), http://www.degruyter.com/view/j/jmc.2014.8.issue-4/jmc-2013-0033/jmc-2013-0033.xml, https://eprint.iacr.org/2013/597
24. Fotiadis, G., Konstantinou, E.: Pairing-friendly elliptic curves resistant to TNFS attacks. 7th International Conference on Algebraic Informatics (June 2017), http://www.icsd.aegean.gr/publication_files/conference/880724856.pdf
25. Fotiadis, G., Martindale, C.: Optimal tnfs-secure pairings on elliptic curves with even embedding degree. Cryptology ePrint Archive, Report 2018/969 (2018), https://eprint.iacr.org/2018/969
26. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology **23**(2), 224–280 (Apr 2010)
27. Fried, J., Gaudry, P., Heninger, N., Thomé, E.: A kilobit hidden SNFS discrete logarithm computation. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 202–231. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_8
28. Gordon, D.M.: Designing and detecting trapdoors for discrete log cryptosystems. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 66–75. Springer, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_5
29. Granger, R., Scott, M.: Faster squaring in the cyclotomic subgroup of sixth degree extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 209–223. Springer, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-13013-7_13
30. Guillevic, A.: Simulating DL computation in GF(p^n) with the new variants of the Tower-NFS algorithm to deduce security level estimates (Nov 2017), slides at ECC 2017 workshop. https://ecc2017.cs.ru.nl/
31. Joux, A.: A one round protocol for tripartite Diffie-Hellman. Journal of Cryptology **17**(4), 263–276 (Sep 2004)
32. Joux, A., Pierrot, C.: The special number field sieve in $\mathbb{F}_{p^n}$ - application to pairing-friendly constructions. In: Cao, Z., Zhang, F. (eds.) PAIRING 2013. LNCS, vol. 8365, pp. 45–61. Springer, Heidelberg (Nov 2014). https://doi.org/10.1007/978-3-319-04873-4_3
33. Karabina, K.: Squaring in cyclotomic subgroups. Math. Comput. **82**(281), 555–579 (2013). https://doi.org/10.1090/S0025-5718-2012-02625-1
34. Khandaker, M.A.A., Nanjo, Y., Ghammam, L., Duquesne, S., Nogami, Y., Kodera, Y.: Efficient optimal ate pairing at 128-bit security level. In: Patra, A., Smart, N.P. (eds.) INDOCRYPT 2017. LNCS, vol. 10698, pp. 186–205. Springer, Heidelberg (Dec 2017)
35. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_20
36. Kiyomura, Y., Inoue, A., Kawahara, Y., Yasuda, M., Takagi, T., Kobayashi, T.: Secure and efficient pairing at 256-bit security level. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17. LNCS, vol. 10355, pp. 59–79. Springer, Heidelberg (Jul 2017). https://doi.org/10.1007/978-3-319-61204-1_4
37. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H.J.J., Timofeev, A., Zimmermann, P.: Factorization of a 768-bit RSA modulus. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 333–350. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_18

38. Kleinjung, T., Diem, C., Lenstra, A.K., Priplata, C., Stahlke, C.: Computation of a 768-bit prime field discrete logarithm. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 185–201. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_7
39. Lewko, A.B.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_20
40. Li, L., Wu, H., Zhang, F.: Pairing computation on edwards curves with high-degree twists. In: Lin, D., Xu, S., Yung, M. (eds.) Inscrypt, Revised Selected Papers. LNCS, vol. 8567, pp. 185–200. Springer, Guangzhou, China (November 27-30 2013). https://doi.org/10.1007/978-3-319-12087-4_12, https://eprint.iacr.org/2012/532
41. Menezes, A., Sarkar, P., Singh, S.: Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. In: Phan, R.C., Yung, M. (eds.) Mycrypt Conference, Revised Selected Papers. LNCS, vol. 10311, pp. 83–108. Springer, Kuala Lumpur, Malaysia (December 1-2 2016). https://doi.org/10.1007/978-3-319-61273-7_5
42. Montgomery, P.L.: Five, six, and seven-term Karatsuba-like formulae. IEEE Transactions on Computers **54**, 362–369 (March 2005). https://doi.org/10.1109/TC.2005.49
43. National Institute of Standards and Technology: Recommendation key management (part 1: General); SP 800-57 Part 1 (2016). https://doi.org/10.6028/NIST.SP.800-57pt1r4, fourth revision
44. Pereira, G.C., Simplício, M.A., Naehrig, M., Barreto, P.S.: A family of implementation-friendly BN elliptic curves. Journal of Systems and Software **84**(8), 1319 – 1326 (2011). https://doi.org/10.1016/j.jss.2011.03.083, http://www.sciencedirect.com/science/article/pii/S0164121211000914
45. Sarkar, P., Singh, S.: A general polynomial selection method and new asymptotic complexities for the tower number field sieve algorithm. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 37–62. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_2
46. Schirokauer, O.: The number field sieve for integers of low weight. Math. Comp. **79**(269), 583–602 (January 2010). https://doi.org/10.1090/S0025-5718-09-02198-X
47. Scott, M.: Implementing cryptographic pairings (invited talk). In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) PAIRING 2007. LNCS, vol. 4575, pp. 177–196. Springer, Heidelberg (Jul 2007). https://doi.org/10.1007/978-3-540-73489-5
48. Semaev, I.A.: Special prime numbers and discrete logs in finite prime fields. Math. Comp. **71**(737), 363–377 (2002). https://doi.org/10.1090/S0025-5718-00-01308-9
49. Sutherland, A.V.: Accelerating the CM method. LMS Journal of Computation and Mathematics **15**, 172–204 (2012). https://doi.org/10.1112/S1461157012001015
50. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (Jan 2010). https://doi.org/10.1109/TIT.2009.2034881
51. Weber, D., Denny, T.F.: The solution of McCurley's discrete log challenge. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 458–471. Springer, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055747

52. Zhang, X., Lin, D.: Analysis of optimum pairing products at high security levels. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 412–430. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_24

A Second part of the final exponentiation for $k = 8$

As an illustration, we give here pseudo-code that raises a finite field element a to the power $c = (p + 1 - t_0)/r$, in the case $k = 8$ and $i = 5$. Recall that since the first part of the final exponentiation has been done, we know that $a^{p^4+1} = 0$ so that $a^{-1} = a^{p^4} = \overline{a}$ where the conjugate is taken over the subfield $\mathbb{F}_{p^4}$. The formula below is specific to $i = 5$, but we let $T = 4U + 2V$ which is the most general form (with $V \in \{0, 1\}$). If we apply this to the parameters in §6.1, we can do some simplifications using $V = 0$ (in square brackets below).

$$
\begin{array}{lr}
a_y = a^y;\ a_u = a^u;\ a_Q = a_y^y a_u^u;\ b = a_Q\overline{a_u}; & (2\mathbf{c}_u + 2\mathbf{c}_y + 2\mathbf{m}_k) \\
b = b^2;\ b = (b^2 a)^U b^V;\ b = b\overline{a_y}; & (\mathbf{c}_T + 2\mathbf{m}_k) \\
b = b^2;\ [b = ba^V];\ b = (b^2)^U b^V;\ b = b\overline{a_y}; & (\mathbf{c}_T + \mathbf{m}_k[+\mathbf{m}_k]) \\
b = b^2;\ b = (b^2 a)^U b^V;\ b = b\overline{a_u};\ b = ba; & (\mathbf{c}_T + 3\mathbf{m}_k) \\
b = b^2;\ [b = ba^V];\ b = (b^2)^U b^V;\ b = ba_Q; & (\mathbf{c}_T + \mathbf{m}_k[+\mathbf{m}_k])
\end{array}
$$

The cost is $11\mathbf{m}_k + 4\mathbf{c}_T + 2\mathbf{c}_u + 2\mathbf{c}_y$ in general, and $2\mathbf{m}_k$ less if $V = 0$, using the notations of §5.2. Here we use $\mathbf{c}_T$ to represent a set of operations whose cost is similar to $b = b^2$ followed by $b = (b^2)^U b^V$, although scheduling above is sometimes different.

B Estimating the cost of NFS, NFS-HD and TNFS

We would like to measure with the same methodology as Barbulescu and Duquesne in [7] the cost of computing a discrete logarithm in $\mathbb{F}_{p_5^5}$, $\mathbb{F}_{p_6^6}$, $\mathbb{F}_{p_7^7}$, and $\mathbb{F}_{p_8^8}$, and compare it with a prime field $\mathbb{F}_{p_1}$ of 3072 bits. In our setting the primes p_1, p_5, p_7 have no structure so we cannot use the Joux–Pierrot (JP) polynomial selection. The primes p_6 and p_8 have some structure but not enough to provide an advantage to the JP method (see §3). We compare the NFS, NFS-HD and TNFS variants. We give the best parameters we have found to minimise the running-time of the relation collection and the linear algebra steps, in the sense of [7]. Moreover we make available all the needed SageMath code to run our experiments (see §1.1)

Contrary to [36], we not only estimate the size of the norms but we generate polynomials for each polynomial selection method available, we find parameters (see §B.2) so that enough relations are obtained, and we compare the estimated cost.

A remark about Special *NFS and TNFS.* We consider that a prime p in our set or primes $(p_1, p_5, p_6, p_7, p_8)$ is special and provides a notable advantage to the SNFS or STNFS algorithm if it can be written $p = P(u)$ where $u \approx p^{1/d}$, and P is a polynomial of degree at least 3 and whose coefficients are much smaller than $p^{1/d}$. As a counterexample, for any prime p we can use the base-m polynomial selection method. It chooses $m = \lfloor p^{1/d} \rceil$, writes p in basis m and outputs the corresponding polynomial P such that $p = P(m)$. Then $\|P\|_\infty = m$, and the coefficients are large.

The vulnerabilities of pairing-friendly curves are the following:

- a special prime p given by a polynomial of degree > 2 and tiny coefficients, for instance $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ for BN curves [26, Ex. 6.8].In that case, the Joux–Pierrot polynomial selection method (SNFS) allows a better complexity of NFS, in $L_{p^k}(1/3, 1.923)$.
- a composite embedding degree k allowing the Kim–Barbulescu variant of TNFS. Note that the original TNFS algorithm where $\deg h = k$ does apply but usually is not efficient when it is not combined with the Special variant.
- We note that the effectiveness of STNFS is not very clear. In [7], the authors found that for the KSS curves with $k = 16$, the optimal choice is $\deg h = k = 16$, which is the original setting of Tower NFS, as in [9]. The key point is the special form of p: the prime is given by a polynomial $p(s) = (s^{10} + 2s^9 + 5s^8 + 48s^6 + 152s^5 + 240s^4 + 625s^2 + 2398s + 3125)/980$.

B.1 Choices of polynomials

Figure 10 shows the polynomials on the NFS setting and in the Tower-NFS setting.

$$\begin{tikzcd}
K_0 = \mathbb{Q}[x]/(f_0(x)) \arrow[dr, dash, "\deg f_0 \geq k"'] & & K_1 = \mathbb{Q}[x]/(f_1(x)) \arrow[dl, dash, "\deg f_1 \geq k"] \\
& \mathbb{Q} &
\end{tikzcd}$$

(a) NFS number fields, $p^k \mid \mathrm{Res}(f_0, f_1)$

$$\begin{tikzcd}
K_0 = K_h[x]/(f_0(x)) \arrow[dr, dash, "\deg f_0 \geq k_2"'] & & K_1 = K_h[x]/(f_1(x)) \arrow[dl, dash, "\deg f_1 \geq k_2"] \\
& K_h = \mathbb{Q}[y]/(h(y)) \arrow[d, dash, "\deg h = k_1"] & \\
& \mathbb{Q} &
\end{tikzcd}$$

(b) Tower-NFS number fields, $k_1 k_2 = k,\ p^k \mid \mathrm{Res}_y(h, \mathrm{Res}_x(f_0, f_1))$

Fig. 10: Extensions of number fields for NFS and Tower variants

Polynomial h. In the Tower-NFS setting, the degree of the polynomial h divides the degree k of the extension. For $k = 6$, we can take $\deg h \in \{2, 3, 6\}$ for example. We search for all monic polynomials h of chosen degree, coefficients in $\{0, 1, -1\}$ to minimise the norms, and of small Dedekind-zeta value $\zeta_{K_h}(2)$, so that its inverse $1/\zeta_{K_h}(2)$ is as close as possible to 1 (in practice, we observed that this value is in the interval $]0.4, 1.0[$).

Polynomials f_0, f_1. These two polynomials are selected according to a polynomial selection method: JLSV$_1$, JLSV$_2$, Joux–Lercier, Generalised Joux–Lercier, Conjugation, Joux–Pierrot (Special case), Sarkar–Singh (see for example [8,32,45]).

B.2 Methodology for cost estimation

Relation collection cost. To estimate this cost, we need first to discuss how relation collection will be performed. We make the conservative assumption that a sieving method can always be used. While this is of course commonplace for NFS computations, the same does not hold for TNFS, which needs $(2\deg h)$-dimensional sieving for tuples of the form $(a_0, \ldots, a_{\deg h-1}, b_0, \ldots, b_{\deg h-1})$. As a consequence of this assumption, the relation collection cost can be approximated as the size of the set of tuples. (Alternatively, relation collection can also use smoothness detection algorithms based on remainder trees, which can perform well in practice, see e.g. [38].)

We estimate the size of the set of tuples $(a_0, \ldots, a_{\deg h-1}, b_0, \ldots, b_{\deg h-1})$ processed in the relation collection of the TNFS algorithm to be

$$S^0_{\mathrm{TNFS}}(h, A) = (2A + 1)^{2\deg h}/2 \tag{4}$$

and its core part (duplicates removed) to be

$$S^1_{\mathrm{TNFS}}(h, A) = (2A + 1)^{2\deg h}/(2w(h)\zeta_{K_h}(2)) \ . \tag{5}$$

We consider that the cost of the relation collection is proportional to the first quantity $S^0_{\mathrm{TNFS}}(h, A)$, and to simplify, we assume that this is $S^0_{\mathrm{TNFS}}(h, A)$. We estimate that the number of unique relations obtained is $S^1_{\mathrm{TNFS}}(h, A)$ times the average smoothness probability. For the NFS-HD algorithm, we estimate the size of the set of tuples $(a_0, \ldots, a_{\dim-1})$ to be

$$S^0_{\mathrm{NFS-HD}}(\dim, A) = (2A + 1)^{\dim}/2 \tag{6}$$

and its core part (duplicates removed) to be

$$S^1_{\mathrm{NFS-HD}}(\dim, A) = (2A + 1)^{\dim}/(2\zeta(\dim)) \ . \tag{7}$$

Again, we consider that the cost of the relation collection is $S^0_{\mathrm{NFS-HD}}(\dim, A)$, and the number of unique relations obtained is $S^1_{\mathrm{NFS-HD}}(\dim, A)$ times the average smoothness probability.

In the NFS algorithm, the elements in the relation collection are pairs of integers (a, b). We need a, b to be coprime: the probability is $1/\zeta(2) = 6/\pi^2 \approx 0.60$. For NFS-HD, the probability that a tuple of random integers $(a_0, \ldots, a_{\dim -1})$ has gcd 1 is $1/\zeta(\dim)$. To avoid duplicates, the leading coefficient is chosen positive ((a, b) and $(-a, -b)$ give the same relation).

The generalisation to pairs of coprime ideals depends on the number field K_h defined by h. The probability that two ideals of K_h are coprime is $1/\zeta_{K_h}(2)$. In practice we observed that it can vary from 0.44 to 0.99. Then as in [7, §5.2], we consider torsion units of K_h (it happens if h is a cyclotomic polynomial). Let w be the index of $\{1, -1\}$ in the group of roots of unity in K_h. If $\mathfrak{q}$ is a prime ideal in K_f, then $u\mathfrak{q}$ is also a prime ideal giving the same relation, where u is any root of unity of K_h. We can detect and avoid the case $u = -1$ but (up to now) there does not exist a way to avoid the other roots of unity. The number of tuples that will contribute to distinct relations is divided by $2w$.

The non-torsion units do not contribute to duplicates: their coefficients being quite large, the coefficients of the ideal $u_1\mathfrak{q}$ overpass the bound A and are not considered in the relation collection.

Average smoothness probability. To compute an average smoothness probability, we took at random 10^6 coprime tuples a of coefficients in $[-A, A]$ and positive leading coefficient (this requires about $\zeta(\dim) \cdot 10^6$ random tuples), resp. 10^6 pairs of coprime ideals of K_h (this requires about $\zeta_{K_h}(2) \cdot 10^6$ random tuples). Then we compute the resultants N_f, N_g on both sides (f and g) and we compute the smoothness probability of that tuple as

$$\Pr(a) = \Pr(N_f \text{ is } B\text{-smooth}) \times \Pr(N_g \text{ is } B\text{-smooth}) \; . \tag{8}$$

We compute the average smoothness probability as the average over all the random unique tuples, that is $10^{-6} \sum_{\text{random } a, \text{ coprime}} \Pr(a)$.

We estimate the smoothness probability on one side with the formula

$$\Pr(N \text{ is } B\text{-smooth}) \approx \delta(u) + (1-\gamma)\frac{\delta(u-1)}{\log N}, \text{ where } u = \frac{\log N + \alpha}{\log B} \tag{9}$$

where $\gamma \approx 0.577$ is Euler's constant, and δ is the Dickman rho function.[5]

Linear algebra cost (filtering, block-Wiedemann). We assume that the input of this step is a set of unique relations. Usually, a certain amount of *excess* is required: there are up to twice more relations than prime ideals involved in the relations (at this point, the matrix would be a vertical rectangle of twice more rows than columns). Before the linear algebra, the relations are processed to produce a dense matrix of good quality, in order to ease the linear algebra step. The *filtering* step removes the singletons (the prime ideals corresponding to

[5] We depart from the conventional notation ρ for the Dickman rho function, to avoid confusion with $\rho = \log p / \log r$.

columns that appear only in one relation). Doing this produces new singletons, so this step is done several times (two to ten times for example). Then a "clique removal" is performed, that also reduces part of the excess. Finally, a *merge* step increases the density of the matrix to some target density, reaching 125 to 200 non-zero entries per row in the recent record computations. The yield of the filtering step varies a lot in the literature: it reduced the size of the set of relations by a factor 9 for the SNFS-1024 DLP record [27], and by a factor 386 for the NFS-768 DLP record [38]

$$c_{\text{filtering,min}} = 9, \quad c_{\text{filtering,max}} = 386 \ .$$

We summarise in Table 11 the parameters of the filtering step for the recent record-breaking integer factorisations and discrete logarithm computations[6,7]. When we were not able to collect the data we put a question mark. Contrary to [7], we propose a different interpretation of the filtering step yield: in our point of view, it is highly software-dependent and cryptanalyst-dependent. Indeed, the low values correspond to records by the `cado-nfs` team, while the high values correspond to Kleinjung et al. record computations (the software being not available in the latter case). At first glance, it seems to be due to software performance differences. To refine this impression, we decided to compare the two integer factorisation records of $2^{1039} - 1$ and $2^{1061} - 1$ by the SNFS algorithm: for $2^{1039} - 1$, Kleinjung et al. have chosen a large prime bound of 2^{36} to 2^{38}, while Childers has chosen the *lower* value 2^{33} for the *larger* integer $2^{1061} - 1$ (Table 11). We can also compare the RSA-220 and RSA-768 record factorisations (220 and 232 decimal digits resp.) and obtain the same conclusion.

In fact, a strategy of oversieving was deployed for the DLP-768 record computation. The large prime bound was increased to 2^{36}, while a bound of 2^{31} could have been enough (but it would have required a much higher effort in the linear algebra step). The ratio of ratios is $386.34/8.84 = 43.7$ and part of it is explained by the factor $2^5 = 32$ in the large prime bound choice. The larger set of relations to feed the filtering step allowed to obtain a matrix of better quality, reducing the linear algebra step. The density of rows seems more under control: from 134 to 200. We choose an upper bound: we assume that the density of a row is

$$\text{weight per row} = 200 \ .$$

We estimate the time of the matrix-vector multiplications in the block-Wiedemann algorithm of the linear algebra step to be

$$(\text{number of rows})^2 \times \mathtt{w} \times (\text{weight per row}) \tag{10}$$

where $\mathtt{w}$ is the word-size of subgroup order (in our case, $\log_2 r = 256$ bits and $\mathtt{w} = 4$ words of 64 bits).

[6] For the RSA-768 record factorisation, we used the corrected value 46.7G instead of 47.7G, according to P. Zimmermann's webpage `https://members.loria.fr/PZimmermann/papers/#rsa768`.

[7] We mention that there was a typo in [7, Table 3]: in the factorisation of $2^{1039} - 1$, there were 66.7M rows after filtering, not 82.8M, and the reduction factor of the filtering step is 143, not 167.

	cado-nfs			Kleinjung et al.			NFS@HOME
record	DL-1024 (S)	RSA-220	DL-180	DL-768	RSA-768	$2^{1039}-1$ (IF)	$2^{1061}-1$ (IF)
references	[27]	[6]	[14]	[38]	[37]	[1]	[17]
year	2017	2016	2014	2017	2010	2007	2012
after sieving							
`lpb`	2^{31}	2^{34}	$2^{29}, 2^{30}$	2^{36}	2^{37} to 2^{40}	2^{36} to 2^{38}	2^{33}
unique rows (R_0)	248.94 M	1.17 G	179.29 M	9.08 G	46.76 G	13.8 G	671 M
prime ideals	210.20 M	1.52 G	82.60 M	5.75 G	11.17 to 82.41 G	5.75 to 21.73 G	787 M
columns (C_0)	210.18 M	1.21 G	81.80 M	5.08 G	35.29 G	?	?
excess (R_0/C_0)	1.184	0.97	2.19	1.79	1.32		
after rm singleton							
rows (R_1)	222.77 M	613 M	–	7.84 G	24.62 G	?	?
columns (C_1)	173.58 M	594 M	–	≤3.77 G	≈ 15 G	?	?
excess (R_1/C_1)	1.28	1.03	–	≥2.07 G	1.64		
ratio R_0/R_1	1.12	1.91	–	1.16	1.90		
a. rm clique+singl							
rows (R_2)	95.93 M	496 M	21.46 M	1528 M	2.46 G	755.7 M	282 M
columns (C_2)	95.93 M	496 M	21.46 M	670 M	1.70 G	594.2 M	?
excess (R_2/C_2)	1.00	1.00	1.00	2.28	1.45	1.27	
weight/row	27.68	18.48	28.84	≤ 22.73	?	?	?
ratio R_0/R_2	2.595	2.36	8.40	5.94	19.02	18.26	2.38
after merge							
rows (R_3)	28.15 M	132 M	7.29 M	23.5 M	192.80 M	66.7 M	90.3 M
columns (C_3)	28.15 M	132 M	7.29 M	23.5 M	192.80 M	66.7 M	90.3 M
weight/row	200	175	150	134	144	143	125
ratio R_2/R_3	3.40	3.76	2.95	65.04	12.75	11.33	3.12
ratio R_0/R_3	8.84	8.86	24.60	386.34	242.55	207	7.43

Table 11: Data from recent record computations. Additional data for the DL-1024, DL-180, and DL-768 computations was collected from their respective authors (the `cado-nfs`-team and T. Kleinjung).The number of prime ideals is computed as `log_integral(lpb0) + log_integral(lpb1)`. the PURGE step removes singletons, cliques, and for `cado-nfs`: removes the excess, for Kleinjung et al. records: removes part of the excess. the MERGE step increases the weight per row, reduces the number of rows and columns, and for Kleinjung et al. records: removes the final excess.