Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves

Aurore Guillevic[1,2]

[1] Laboratoire Chiffre – Thales Communications and Security
4 avenue des Louvresses – 92622 Gennevilliers Cedex – France
[2] Crypto Team – DI – École Normale Supérieure
45 rue d'Ulm – 75230 Paris Cedex 05 – France
aurore.guillevic@ens.fr

Abstract. We provide software implementation timings for pairings over composite-order and prime-order elliptic curves. Composite orders must be large enough to be infeasible to factor. They are modulus of 2 up to 5 large prime numbers in the literature. There exists size recommendations for two-prime RSA modulus and we extend the results of Lenstra concerning the RSA modulus sizes to multi-prime modulus, for various security levels. We then implement a Tate pairing over a composite order supersingular curve and an optimal ate pairing over a prime-order Barreto-Naehrig curve, both at the 128-bit security level. We use our implementation timings to deduce the total cost of the homomorphic encryption scheme of Boneh, Goh and Nissim and its translation by Freeman in the prime-order setting. We also compare the efficiency of the unbounded Hierarchical Identity Based Encryption protocol of Lewko and Waters and its translation by Lewko in the prime order setting. Our results strengthen the previously observed inefficiency of composite-order bilinear groups and advocate the use of prime-order group whenever possible in protocol design.

Keywords: Tate pairing, optimal ate pairing, software implementation, composite-order group, supersingular elliptic curve, Barreto-Naehrig curve.

1 Introduction

Bilinear structures of composite-order groups provide new possibilities for cryptosystems. In 2005, Boneh, Goh and Nissim [7] introduced the first public-key homomorphic encryption scheme using composite-order groups equipped with a pairing. The scheme permits several homomorphic additions and one multiplication on few bits. The security relies on the subgroup decision assumption. They applied this tool to on-line voting and universally verifiable computation. Decryption time grows exponentially w.r.t. the input size so this approach for homomorphic encryption is not very practical for large data but the idea was developed for other interests. In the last seven years, many cryptographic schemes

were built using composite-order groups. In 2005, an interesting Hierarchical Identity Based Encryption (HIBE) was proposed by Boneh, Boyen and Goh [6]. It relies on the ℓ-bilinear Diffie-Hellman exponent assumption. In 2009, Waters introduced the Dual System Encryption method [24], resulting in very interesting properties for security proofs. In 2011, Lewko and Waters published a HIBE relying on the subgroup decision assumption. HIBE has become very practical in the sense that the maximal hierarchy depth is not static i.e. can be augmented without resetting all the system parameters.

The subgroup decision assumption is that given a group G of composite order $p_1p_2 = N$ (e.g. an RSA modulus), it is hard do decide whether a given element $g \in G$ is in the subgroup of order p_1 without knowing p_1 and p_2. N must be infeasible to factor to achieve this hardness. This results in very large parameter sizes, e.g. $\log_2 N = 3072$ or 3248 for a 128-bit security level, according to NIST or ECRYPT II recommendations. Moreover, the pairing computation is much slower in this setting but exact performances were not given yet. To reduce the parameter sizes, Freeman [10] proposed to use a copy of the (e.g. 256-bit) same prime-order group instead of a group whose order (of e.g. 3072 bits) has two or more distinct primes. His paper provides conversions of protocols and in particular of the BGN scheme, from the composite-order to the prime-order setting. Then Lewko at Eurocrypt 2012 [19] provided a generic conversion. These conversions achieve much smaller parameter sizes but have a drawback: they need not only one but several pairings. More precisely, Lewko's conversion for the HIBE scheme needs at least $2n$ pairings over a prime order group (of e.g. 256-bit) instead of one pairing over a composite order group (of e.g. 3072-bit) of n primes.

The translated protocols remain interesting because it is commonly assumed that a pairing is much slower over a composite-order than a prime-order elliptic curve. An overhead factor around 50 (at an estimate attributed to Scott) was given in [10, §1] for a 80-bit security level. A detailed and precise comparison would be interesting and useful to protocol designers and application developers.

The Number Field Sieve (NFS) algorithm is the fastest method to factor a two prime modulus. Lenstra studied carefully its complexity and made recommendations. Lenstra stated that at a 128-bit security level, an RSA modulus can have no more than 3 prime factors of the same size, 4 factors at a 192-bit level and 5 at a 256-bit level [17, §4]. We complete his work to obtain the modulus sizes with more than two prime factors, at these three security levels. We then find supersingular elliptic curves of such orders and benchmark a Tate paring over these curves. We also implemented an optimal ate pairing over a prime-order Barreto-Naehrig curve, considered as the fastest pairing (at least in software). With these timings, we are able to estimate the total cost of the protocols in composite-order and prime-order settings. We then compare the BGN protocol [7] in the two settings and do the same for the unbounded HIBE protocol of Lewko and Waters [20] and its translation [19, §B].

Organisation of the paper. Section 2 presents our results on the modulus sizes with more than two prime factors, at the 128, 192 and 256-bit security level. In

Sec. 3, we present the possibilities to construct pairing-friendly elliptic curves of composite order and our choice for the implementation. We develop a theoretical estimation of each pairing in Sec. 4. Our implementation results are presented in Sec. 5.

2 Parameter sizes

In this section, we extend Lenstra's estimates [17] to RSA modulus sizes with up to 8 prime factors. We present in Tab. 1 the usual key length recommendations from `http://www.keylength.com`. The NIST recommendations are the less conservative ones. A modulus of length 3072 is recommended to achieve a security level equivalent to a 128 bit symmetric key. The ECRYPT II recommendations are comparable: 3248 bit modulus are suggested.

Table 1. Cryptographic key length recommendations, January 2013. All key sizes are provided in bits. These are the minimal sizes for security.

Method	Date	Sym-metric	Asymmetric	Discrete Log		Elliptic curve	Hash function
				Key	Group		
Lenstra / Verheul	2076	129	6790–5888	230	6790	245	257
Lenstra Updated	2090	128	4440–6974	256	4440	256	256
ECRYPT II (EU)	2031–2040	128	3248	256	3248	256	256
NIST (US)	> 2030	128	3072	256	3072	256	256
FNISA (France)	> 2020	128	4096	200	4096	256	256
NSA (US)	–	128	–	–	–	256	256
RFC3766	–	128	3253	256	3253	242	–

We consider the Number Field Sieve attack (NFS, see e.g. [18] for an overview) whose complexity is given by [17, §3.1]:

$$L[N] = \exp(1.923(\log N)^{1/3}(\log\log N)^{2/3}) \text{ (NFS)} \tag{1}$$

and the Elliptic Curve Method (ECM) that depends on the modulus size and on the size of the smallest prime p_i in the modulus. This attack is less efficient for a modulus of only two prime factors but become competitive for more prime factors. We consider that all the prime factors p_i have the same size. The ECM complexity is [17, §4]

$$E[N, p_i] = (\log_2 N)^2 \exp(\sqrt{2}(\log p_i)^{1/2}(\log\log p_i)^{1/2}) \text{ (ECM)}. \tag{2}$$

It is assumed in [17, §3.1] that a k-bit RSA modulus offers the same *computational security* as a symmetric cryptosystem of d-bit security and speed comparable to singe DES if $L[2^k] = 50 \cdot 2^{d-56} \cdot L[2^{512}]$. They argue that speed-up in symmetric implementation affects slightly the complexity thus is not taken into account. We used this formula to compute Tab. 2. These assumptions may be

considered controversial, anyone can consider more conservative ones and obtain slightly different results. Results are presented in Tab. 2.

The first line in Tab. 2 appears in [17, Tab. 1]. The threshold between NFS and ECM is represented through bold font. We do not consider security levels under 128 bits. For a 128-bit security level, a modulus of 3224 bits with two prime factors (of 1612 bits) is enough to prevent the NFS attack and the attack with ECM is much slower. This attack becomes significantly more efficient than the NFS one against a modulus with 5 prime factors (each of the same size). A modulus of 4040 bits instead of 3224 bits must be considered. For 8 primes in the modulus, the size is almost doubled: 6344 bits instead of 3224 bits and each prime factor is 793-bit long. Table 2 could be used by protocol designers to set the size of the security parameter λ. Our Tab. 2 can also be used when setting the parameter sizes for protocols (or security proofs) relying on the Φ-hiding assumption. In 2010 at Crypto, Kiltz, O'Neill and Smith [15] used this assumption to obtain a nice result about RSA-OAEP. Then at Africacrypt in 2011, Herrmann [13] explained new results about the security of this assumption. We emphasise that setting the security parameter λ in protocols is not completely straightforward if the modulus contains more than 3 prime factors.

Table 2. RSA-Multi-Prime modulus size from 2 (see [17, Tab. 1]) up to 8 prime factors

Equiv.	AES-128				AES-192				AES-256			
Nb of	min		max		min		max		min		max	
primes	$\log p_i$	$\log N$	$\log p_i$	$\log N$	$\log p_i$	$\log N$	$\log p_i$	$\log N$	$\log p_i$	$\log N$	$\log p_i$	$\log N$
2	1322	2644	1612	3224	3449	6898	3959	7918	6920	13840	7694	15388
3	882	2646	1075	3225	2299	6897	2640	7920	4614	13842	5129	15387
4	694	**2776**	815	**3260**	1725	6900	1980	7920	3460	13840	3847	15388
5	687	**3435**	808	**4040**	1484	**7420**	1654	**8270**	2768	13840	3078	15390
6	682	**4092**	802	**4812**	1476	**8856**	1646	**9876**	2544	**15264**	2760	**16560**
7	677	**4739**	797	**5579**	1470	**10290**	1639	**11473**	2535	**17745**	2752	**19264**
8	673	**5384**	793	**6344**	1464	**11712**	1633	**13064**	2528	**20224**	2744	**21952**

3 Composite-order elliptic curves

For a detailed introduction to pairings, see e.g. [14, Ch. IX]. Let E be an elliptic curve defined over a prime field $\mathbb{F}_p$. A pairing is a bilinear, non-degenerate and efficient map $e : G_1 \times G_2 \to G_T$. From an algebraic point of view, G_1 and G_2 are two *necessarily distinct* subgroups of $E(\bar{\mathbb{F}}_p)$, of same order n. If $n \mid \#\quad(\mathbb{F}_p)$, $G_1 \subset E(\mathbb{F}_p)$, this is the common setup. Let k be the smallest integer such that $n \mid p^k - 1$, k is the *embedding degree*. Then $G_2 \subset E(\mathbb{F}_{p^k})$ and $G_T \subset \mathbb{F}^*_{p^k}$, except for $k = 1$. For supersingular or some of the $k = 1$ curves, an efficient isomorphism is available from G_1 into G_2. This gives a symmetric pairing and we can use the notation $G_1 = G_2$ to implicitly denote the use of the isomorphism in the

pairing computation. In the remaining of this section, we will use the algebraic interpretation of G_1 and G_2. In other words, we will assume that they are two distinct subgroups of E, of same order n. The target group G_T is the order-n (multiplicative) subgroup of $\mathbb{F}_{p^k}^*$. G_1 and G_2 have to be strong enough against a generic attack to a discrete logarithm problem. The third group G_T is more vulnerable because computing a discrete logarithm in a finite field is easier with the index calculus attack. Its size has to be enlarged.

Finding optimal pairing-friendly elliptic curves is an active field of research (see the survey [11]). At a 128-bit security level, the optimal choice would be to construct an elliptic curve whose order is a prime of 256 bits and over a prime finite field of the same size. For an embedding degree $k = 12$, an element in the third group is 3072 bit long in order to match the NIST recommendations. Such optimal pairing-friendly curves exist [3] (Barreto-Naehrig (BN) curves), but have a special form: the parameters p (defining the finite field), n (elliptic curve order) and t (trace) are given by degree 4 polynomials. We have $p(x) = 36x^4+36x^3+24x^2+6x+1$, $n(x) = 36x^4+36x^3+18x^2+6x+1$ and $t(x) = 6x^2+1$.

3.1 Issues in composite-order elliptic curve generation

For our particular purpose, the pairing-friendly elliptic curve order has to contain a composite-order modulus N. Hence the order is chosen *before* the other curve parameters and no special form can be imposed to N. For example, finding such an elliptic curve over a non-prime field (e.g. in characteristic 2 or 3) is completely infeasible at the moment. As for BN curves, all the complete pairing-friendly elliptic curve families in the survey [11], defined by polynomials, are not convenient.

The parameter sizes of composite-order elliptic curve are not optimal. The curve order should be hN with h a cofactor as small as possible. Due to the Hasse bound, the size of p (defining $\mathbb{F}_p$) is the same as the size of hN. This means that the prime field $\mathbb{F}_p$ already achieves the recommended size (say, 3072) to be strong enough against an index calculus attack. Consequently, an embedding degree $k = 1$ is enough. As G_1 and G_2 are distinct, an embedding degree of 1 means that both G_1 and G_2 are subgroups of $E(\mathbb{F}_p)$, then $N^2 \mid E(\mathbb{F}_p)$ and $\log_2 p \geqslant 2 \log_2 N$. This mean that for a 3072 bit modulus N, p will have more than 6144 bits. Such curves exist, for example see [16, §6] or more recently, [8]. The elliptic curve point coordinates are more than 6144 bit long.

Tate pairing computation is described in Alg. 2. It consists in a Miller loop over the considered elliptic curve group order. A final exponentiation in $\mathbb{F}_{p^k}^*$ at the end is performed to obtain a unique pairing value. Optimal ate pairing computation on a BN curve is detailed in Alg. 1. Convenient supersingular curves do not benefit from pairing optimisation such as η_T pairing, as the trace is zero (in large characteristic), or decomposition of the Miller loop length, as there is no efficiently computable endomorphism over $\mathbb{F}_p$ on such curves, except the scalar multiplication. For ordinary curves with $6 \mid k$ and $D = 3$ (BN curves) or $4 \mid k$ and $D = 1$, the complex multiplication induces an easy computable endomorphism thus permits to reduce the Miller loop length up to a factor 4.

Pairing computation over curves of embedding degree 2 needs multiplications over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ with $\log_2 p = 1536$. Pairing computation over curves of embedding degree 1 needs multiplications over $\mathbb{F}_p$ with $\log_2 p = 3072$. Recently in [26] it was shown that self-pairings on these particular curves may be speed-up thanks to the distortion map. Zhao et. al. gave efficient formulas of Weil pairing with denominator elimination thanks to the distortion map, although $k = 1$ instead of $k = 2$. Such ordinary $k = 1$ curves with efficient endomorphisms are rare. Few constructions are proposed in [8]. More work is needed to determine if pairings on these curves are competitive with $k = 2$ curves.

As mentioned in recent works, some properties (cancelling, projecting) are achieved with only composite-order elliptic curves or only asymmetric pairings. More precisely, at Asiacrypt 2012, Seo [22] presented results on the impossibility of projecting pairings in certain cases. Then the only possible instantiation is to choose an ordinary composite-order elliptic curve. Such constructions are possible, see e.g. Boneh, Rubin and Silverberg paper [8] but this seems to be the worst case in terms of parameter sizes and efficiency.

3.2 Our choices

If we want to reduce the size of p (hence of G_1), we can choose a supersingular elliptic curve of embedding degree $k = 2$. This means that $G_1 \subset E(\mathbb{F}_p)$, $G_2 \not\subseteq E(\mathbb{F}_p)$ and both G_1 and G_2 are subgroups of $E(\mathbb{F}_{p^2})$.

$$\begin{array}{r|l} G_1 \text{ and } G_2 \subset E(\mathbb{F}_{p^2}) & N^2 \mid \#E(\mathbb{F}_{p^2}) \\ | & \\ G_1 \subset E(\mathbb{F}_p) & N \mid \#E(\mathbb{F}_p),\ N^2 \nmid \#E(\mathbb{F}_p) \end{array}$$

A supersingular elliptic curve of given subgroup order and embedding degree 2 is easy to construct:

1. Let N be a composite-order modulus.
2. Find the smallest integer h, $4 \mid h$, such that $hN - 1$ is prime.
3. Let $p = hN - 1$. The elliptic curve $E(\mathbb{F}_p) : y^2 = x^3 - x$ is supersingular, of order $hN = p + 1$ and embedding degree 2.

As $p = 3 \mod 4$, -1 is not a square in $\mathbb{F}_p$. If $\mathbb{F}_{p^2} = \mathbb{F}_p[Z]/(Z^2 + 1)$, a distortion map is available: $\phi : E(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^2})$, $(x, y) \mapsto (-x, Zy)$. In particular, $\phi(G_1) = G_2$ and the pairing is symmetric. As mentioned above, the improved pairing variant denoted η_T is not possible as this supersingular curve has trace 0 ($\#E(\mathbb{F}_p) = p+1$). We implemented a Tate pairing on this curve. The parameter sizes for a security level equivalent to AES-128 are summarised in Tab. 3. We assume that the points on the elliptic curves are in compressed representation.

4 Theoretical estimation

In this section we will estimate the number of multiplications over the base field for each pairing in Tab. 3.

Table 3. Parameter sizes for prime order and composite order pairing-friendly elliptic curves, minimum and maximum in theory, according to Tab. 2

Elliptic curve, order			size of G_1 order $\log_2 N$ min – max	size of elts in G_1 $\log_2 p$ min – max	emb. deg. k	size of elts in G_2	size of elts in G_T $k \log_2 p$ min – max
BN, prime order			256	256 – 269	12	512 – 538	3072 – 3224
supersingular curve	Prime order		256	1322 – 1612	2	As for elts in G_1	2644 – 3224
	Composite order	2 primes	2644 – 3224	$\geqslant$ 2646 – $\geqslant$ 3226			$\geqslant$ 5292 – $\geqslant$ 6452
		3 primes	2646 – 3225	$\geqslant$ 2648 – $\geqslant$ 3227			$\geqslant$ 5296 – $\geqslant$ 6454
		4 primes	2776 – 3260	$\geqslant$ 2778 – $\geqslant$ 3262			$\geqslant$ 5556 – $\geqslant$ 6524
		5 primes	3435 – 4040	$\geqslant$ 3437 – $\geqslant$ 4042			$\geqslant$ 6874 – $\geqslant$ 8084
		6 primes	4092 – 4812	$\geqslant$ 4094 – $\geqslant$ 4814			$\geqslant$ 8188 – $\geqslant$ 9628
		7 primes	4739 – 5579	$\geqslant$ 4741 – $\geqslant$ 5581			$\geqslant$ 9482 – $\geqslant$ 11162
		8 primes	5384 – 6344	$\geqslant$ 5386 – $\geqslant$ 6346			$\geqslant$ 10772 – $\geqslant$ 12692

4.1 Prime order BN curve

We aim to implement a state of the art optimal ate pairing on a BN curve. We use various techniques described e.g. in [21,5]. A careful operation count is detailed in Alg. 1 since it may be of independent interest. We use the finite field arithmetic described in [9] and [12] for speeding up the pairing final exponentiation and exponentiations in G_T. Operation counts in Tab. 4 describe our choices according to recommendations made in [9]. The arithmetic operations in $\mathbb{F}_p$ are denoted M_p for a multiplication, S_p for a square, I_p for an inversion and HW denotes the Hamming weight. For exponentiation in $\mathbb{F}_{p^k}$, $S_{\Phi_6(p^2)}$ denotes the improved squaring formula from [12]. Details are provided in Alg. 1 which

Table 4. Approximation of arithmetic operations in finite field extensions

$M_{p^{12}} = 3M_{p^6} + 5A_{p^6} + 1M_Y$	$\to 54M_p$	$S_{p^{12}} = 2M_{p^6} + 4A_{p^6} + 2M_Y$	$\to 36M_p$
$M_{p^6} = 6M_{p^2} + 13A_{p^2} + 2M_\beta$	$\to 18M_p$	$S_{p^6} = 2M_{p^2} + 3S_{p^2} + 10A_{p^2} + 2M_\beta$	$\to 12M_p$
$M_{p^2} = 3M_p + 5A_p + 1M_\alpha$	$\to 3M_p$	$S_{p^2} = 2M_p + 4A_p + 2M_\alpha$	$\to 2M_p$

computes $e_{\mathrm{OptAte}}(P, \psi_6(Q)) = f^{\frac{p^{12}-1}{r}}$ with
$f = f_{6x+2,\psi_6(Q)}(P) \cdot \ell_{[6x+2]\psi_6(Q),\pi_p(\psi_6(Q))}(P) \cdot \ell_{[6x+2]\psi_6(Q)+\pi_p(\psi_6(Q)),-\pi_p^2(\psi_6(Q))}(P)$
with ψ_6 the sextic twist map, π_p the p-power Frobenius and π_{p^2} the p^2-power Frobenius.

4.2 Supersingular curve

A Tate pairing may not benefit from the previous optimisations. We can still simplify the Miller loop thanks to the even embedding degree ($k = 2$). The denominators cancel in the final exponentiation thus we can remove them in the

Algorithm 1: Optimal ate pairing $e_{\mathrm{OptAte}}(P,\psi_6(Q))^{\frac{p^{12}-1}{n}}$ on a BN curve

Input: $E(\mathbb{F}_p)$, $P(x_P,y_P)\in E(\mathbb{F}_p)[n]$, $Q(x_Q,y_Q)\in E'(\mathbb{F}_{p^2})[n]$, t, x
Output: $e_{\mathrm{OptAte}}(P,\psi_6(Q))\in\boldsymbol{\mu}_n\subset\mathbb{F}^*_{p^{12}}$

1	$R(X_R:Y_R:Z_R)\leftarrow(x_Q:y_Q:1)$; $f\leftarrow 1$; $s\leftarrow 6x+2$	
2	**for** $m\leftarrow\lfloor\log_2(s)\rfloor-1,\ \ldots,\ 0$ **do**	
3	$(R,\ell)\leftarrow g(R,P)$	$6M_{p^2}+5S_{p^2}+4M_p=32M_p$
4	$f\leftarrow f^2\cdot\ell$	$S_{p^{12}}+13M_{p^2}=36+39=75M_p$
5	**if** $s_m=1$ **then**	
6	$(R,\ell)\leftarrow h(R,Q,P)$	$10M_{p^2}+3S_{p^2}+4M_p=40M_p$
7	$f\leftarrow f\cdot\ell$	$13M_{p^2}=39M_p$
8	$Q_1\leftarrow\pi_p(Q)$	$M_{p^2}=3M_p$
9	$Q_2\leftarrow\pi_{p^2}(Q)$	$2M_p$
10	$(R,\ell)\leftarrow h(R,Q_1,P)$	$6M_{p^2}+5S_{p^2}+4M_p=32M_p$
11	$f\leftarrow f\cdot\ell$	$13M_{p^2}=39M_p$
12	$(R,\ell)\leftarrow h(R,Q_2,P)$	$6M_{p^2}+5S_{p^2}+4M_p=32M_p$
13	$f\leftarrow f\cdot\ell$	$13M_{p^2}=39M_p$
		`1. 8 to 1. 13:` $147M_p$
		`Miller Loop:` $147M_p+\log_2(6x+2)\cdot 107M_p+\mathrm{HW}(6x+2)\cdot 79M_p$
14	$f\leftarrow f^{p^6-1}$	$3M_{p^6}+2S_{p^6}+10M_{p^2}+3S_{p^2}+2M_p+2S_p+I_p=118M_p+I_p$
15	$f\leftarrow f^{p^2+1}$	$10M_p+M_{p^{12}}=64M_p$
16	**if** $x<0$ **then**	
17	$a\leftarrow f^{6\lvert x\rvert-5}$	$\log_2(6x+5)S_{\Phi_6(p^2)}+\mathrm{HW}(6x+5)M_{p^{12}}$
18	**else** $(f^{p^6}=f^{-1})$	
19	$a\leftarrow(f^{p^6})^{6x+5}$	
20	$b\leftarrow a^p$	$5M_{p^2}=15M_p$
21	$b\leftarrow ab$	$M_{p^{12}}=54M_p$
22	Compute f^p, f^{p^2} and f^{p^3}	$5M_{p^2}+10M_p+5M_{p^2}=40M_p$
23	$c\leftarrow b\cdot(f^p)^2\cdot f^{p^2}$	$S_{\Phi_6(p^2)}+2M_{p^{12}}=126M_p$
24	$c\leftarrow c^{6x^2+1}$	$\log_2(6x^2+1)S_{\Phi_6(p^2)}+\mathrm{HW}(6x^2+1)M_{p^{12}}$
25	$f\leftarrow f^{p^3}\cdot c\cdot b\cdot(f^p\cdot f)^9\cdot a\cdot f^4$	$7M_{p^{12}}+5S_{\Phi_6(p^2)}=468M_p$
		`Exponentiation` $f\leftarrow f^{(p^6-1)(p^2+1)(p^4-p^2+1)/n}$:
		$(885+18\log_2(6x+5)+54\mathrm{HW}(6x+5)+18\log_2(6x^2+1)+54\mathrm{HW}(6x^2+1))M_p+I_p$
26	**return** f	

computations. Details are provided in Alg. 2 with ψ_2 the distortion map from G_1 into G_2.

The algorithm for a supersingular elliptic curve of composite order is the same as Alg. 2. In addition, we take $n=N$ the modulus, hence $\log_2 n=3072$ for example. By construction, the cofactor h will be as small as possible, resulting in very cheap final exponentiation, e.g. $\log_2 h=12$. We detail in Tab. 5 the different estimations for a pairing computation.

Algorithm 2: Tate pairing $e_{\mathrm{Tate}}(P,\psi_2(Q))^{\frac{p^2-1}{n}}$ on a supersingular curve

Input: $E(\mathbb{F}_p): y^2 = x^3 + ax$, $P(x_P, y_P), Q(x_Q, y_Q) \in E(\mathbb{F}_p)[n]$, n
Output: $e_{Tate}(P,\psi_2(Q)) \in \boldsymbol{\mu}_n \subset \mathbb{F}^*_{p^2}$

1 $R(X_R : Y_R : Z_R) \leftarrow (x_P : y_P : 1)$; $f \leftarrow 1$; **for** $m \leftarrow \lfloor \log_2(n) \rfloor - 1, \ldots, 0$ **do**
2 $\quad (R,\ell) \leftarrow g(R,Q)$ — $8M_p + 6S_p$
3 $\quad f \leftarrow f^2 \cdot \ell$ — $S_{p^2} + M_{p^2} = 5M_p$
4 $\quad$ **if** $n_m = 1$ **then**
5 $\quad\quad (R,\ell) \leftarrow h(R,P,Q)$ — $11M_p + 3S_p$
6 $\quad\quad f \leftarrow f \cdot \ell$ — $M_{p^2} = 3M_p$

Miller loop: $\log_2 n \cdot (13M_p + 6S_p) + \mathrm{HW}(n) \cdot (14M_p + 3S_p)$

7 $f \leftarrow f^{p-1}$ — $2M_p + I_p$
8 $f \leftarrow f^{(p+1)/n} = f^h$ — $\log_2 h \;\; S_{p^2} + \mathrm{HW}(h) M_{p^2}$
9 **return** f

Final exp.: $\log_2 h \;\; S_{p^2} + \mathrm{HW}(h) M_{p^2} + 2M_p + I_p$

Table 5. Estimations for pairings on prime-order and composite-order elliptic curves, assuming that for a composite-order supersingular curve, $\log_2 N$ is as in Tab. 2, $\mathrm{HW}(N) = \log_2 N/2$, $\log_2 h = 12$ and $\mathrm{HW}(h) = 5$, and for a BN curve, $\log_2 n = \log_2 p = 256$, $\mathrm{HW}(x) = 4, \mathrm{HW}(6x+5) = 10, \mathrm{HW}(6x^2+1) = 33$.

Curve	Pairing	nb primes	Miller loop min – max	Final exp. (+ I_p) min – max
BN	opt. ate	1	7204 M_p	6669 M_p
supersingular (SsC)	Tate	1	$4224M_p + 1728S_p$	$3730M_p$ – $4745M_p$
		2	$52880M_p + 19830S_p$– $64480M_p + 24180S_p$	$41M_p + I_p$
		3	$52920M_p + 19845S_p$– $64500M_p + 24187S_p$	
		4	$55520M_p + 20820S_p$– $65200M_p + 24450S_p$	
		5	$68700M_p + 25762S_p$– $80800M_p + 30300S_p$	
		6	$81840M_p + 30690S_p$– $96240M_p + 36090S_p$	
		7	$94780M_p + 35542S_p$–$111580M_p + 41842S_p$	
		8	$107680M_p + 40380S_p$–$126880M_p + 47580S_p$	

5 Implementation results

We implemented in C the above pairings (Tab. 3), we compiled with gcc 4.4.3 and ran the software implementation on a 2.6 GHz Intel Celeron 64 bits PC with 1 GB RAM and Ubuntu 10.04.4 LTS OS. The developed code is part of a proprietary library, the LibCryptoLCH developed at Thales Communications & Security (France). The finite field arithmetic uses the Montgomery representation and the modular multiplication is written in x86-64 assembly language. Our timings are competitive compared to others proprietary generic libraries such as the one used at Microsoft Research [1]. In this paper, They develop a C library then add different optimised assembly part of code for x86 and ARMv7 processors. They run their library on a x86-64, Intel Core2 E6600 @ 2.4 GHz, Windows 7 (64-bit) and on a ARM, dual-core Cortex A9 @ 1GHz, Windows device. They obtain a pairing on average at 55.19 ms (ARM) and 6.31 ms (x86-64) in projective

coordinates and 51.01 ms (ARM) and 5.92 ms (x86-64) in affine coordinates, over a BN curve of 254 bit prime order group. Our timings are slightly slower than other state-of-the-art ones can be ([21,2]) because our software is not optimised for a particular sparse prime number which might result in very specific and optimised modular reduction.

Results are presented in Fig. 1. We did not plot our timings on a BN curve as the spots would be on the x axis because of the scale. We present in Tab. 6 our results for a BN curve, a prime-order and a composite two-prime order supersingular curve. The first line shows our results of an implementation of an optimal ate pairing on a Barreto-Naehrig curve, see for example [23,5,21] on how to implement it efficiently. We choose a quite sparse but still random parameter $x =$ `0x580000000000100d` resulting in quite sparse prime order and prime field. Our modular reduction is not optimised for this value. Our extension field is optimised for towers built with binomials with small coefficients. For instance the first extension is built as $\mathbb{F}_{p^2} \simeq \mathbb{F}_p[X]/(X^2+1)$ as $p \equiv 3 \mod 4$ which allows a fast reduction $\mod X^2+1$ in the Karatsuba multiplication. The second extension is built as $\mathbb{F}_{p^{12}} \simeq \mathbb{F}_{p^2}[Y]/(Y^6-2)$ resulting in fast polynomial reduction too. Our implementation perform a pairing in 5.05 ms in average which is comparable to the 5.73 ms over an x86-64 Intel Core2 E6600 of the Microsoft Research Team [1, Tab.2].

Table 6. Timings for exponentiation in milliseconds (ms), Ate and Tate pairings on prime order n and composite order $n = n_1 \cdots n_i$ elliptic curves for different security levels.

Pairing	$\log_2 n$	$\log_2 n_i$	$\log_2 p$	$k\cdot \log_2 p$	Miller Loop	F. Exp.	Pairing	Exp. G_1	g^{p_i} G_1	Exp. G_2	Exp. G_T	g^{p_i} G_T
BN,o.ate	256	–	256	3072	2.35	2.70	5.05	0.55	–	1.91	5.16	–
	269	–	269	3228	3.22	3.80	7.29	0.77	–	2.56	5.98	–
(1), Tate	256	–	1536	3072	19.70	20.50	40.20	8.30	–	–	2.20	–
(2), Tate	1024	512	1036	3072	56.88	0.10	56.98	24.38	13.12	–	7.81	3.9
(2), Tate	2048	1024	2059	4118	392.50	0.40	392.90	172.5	86.25	–	50.63	25.8
(2), Tate	3072	1536	3083	6166	1295.6	0.7	1296.3	586.2	301.8	–	166.10	81.9
(3), Tate	3072	1024	3083	6166	1275.6	0.7	1276.3	556.9	222.5	–	174.88	60.1

Last year Zhang et al. in [25] published an optimised implementation of composite-order bilinear pairings on GPU. They obtained a very efficient execution time of 17.4 ms, 11.9 ms and 8.7 ms per pairing in average with a 1024 bit modulus on three different GPU [25, §8]. With PBC library on an Intel Core 2 E8300 CPU at 2.83 GHz and 3GB RAM they obtained 171.1 ms. With our library on an Intel Celeron as specified above, we obtain 57 ms for a pairing over a 1024 bit modulus order elliptic curve and 393 ms for a 2048 bit modulus order. This means our library is two times faster than PBC in this setting, mostly because of our x86-64 implementation of the multiplication in $\mathbb{F}_p$.

For this 128-bit security level, a pairing on an elliptic curve of composite order with two primes is 254 times slower than over a prime-order elliptic curve (1.27 s compared to 5.05 ms). The Miller loop is very expensive, indeed it runs

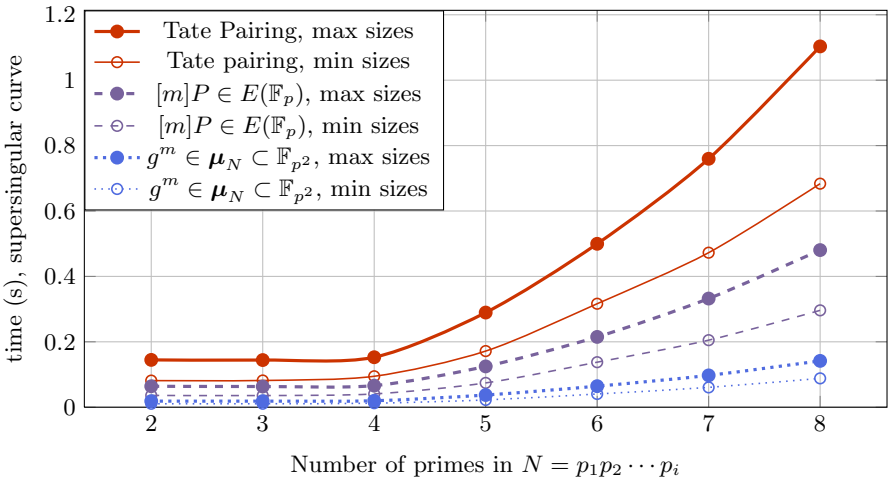

Fig. 1. Execution time on average for a scalar multiplication on $E(\mathbb{F}_p)$, an exponentiation in $\boldsymbol{\mu}_N \subset F_{p^2}$ and a Tate pairing over a composite-order supersingular curve.

over N without any possible optimisation as explained in Sec. 3.1. The final exponentiation is very cheap because it consists in $f^{(p-1)h} = (f^p \cdot f^{-1})^h$ computed with one inversion, one multiplication, one Frobenius map and one very small exponentiation (h is only a dozen bits) in $\mathbb{F}_{p^2}$.

5.1 Application to BGN cryptosystem

In 2005, Boneh, Goh and Nissim published in [7] a somewhat homomorphic encryption scheme which can add several times different ciphertexts, perform one multiplication then continue to add ciphertexts. Freeman proposed a conversion in a prime-order setting in [10]. We compare the two settings. Our results show that the whole protocol is much slower on a composite-order elliptic curve, as presented in Tab. 7.

Protocol Setup(τ)

1. Generate two random τ-bit primes p_1, p_2 and set $N = p_1 p_2$.
2. Generate a (symmetric) bilinear pairing $e : G_1 \times G_1 \to G_T$ with G_1 and G_T of order N.
3. Pick two random generators $g_1, u_1 \leftarrow G_1$ and set $u_{1(p_1)} = u_1^{p_2} \Rightarrow u_{1(p_1)}$ is a random generator of the subgroup of order p_1 of G_1. We denote by $G_{1(p_1)}$ this subgroup. Set $g_T = e(g_1, g_1)$ as generator of G_T and $h_T = e(g_1, u_{1(p_1)}) = g_T^{p_2}$ as generator of the subgroup $G_{T(p_1)}$ of order p_1 of G_T.
4. $\mathcal{PK} = (N, G_1, G_T, e, g_1, u_{1(p_1)}, g_T, h_T)$. $\mathcal{SK} = p_1$.

Encrypt$(\mathcal{PK}, m)$**:** $m \in \mathbb{N}, m < p_2$. Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$. The ciphertext is $c = g_1^m \cdot u_{1(p_1)}^r \in G_1$.
Homomorphic Addition (c_1, c_2) **mod** N**,** $\in G_1$**.** Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$. $c = c_1 \cdot c_2 \cdot u_{1(p_1)}^r = g_1^{m_1+m_2 \mod N} \cdot u_{1(p_1)}^{r'} \in G_1$.
Decrypt$(\mathcal{SK}, c \in G_1)$**:** We have $c^{p_1} = (g_1^m \cdot u_{1(p_1)}^r)^{p_1} = (g_1^{p_1})^m$. Compute the discrete log of c^{p_1} in base $g_1^{p_1}$. This is very slow or m must be very small (few bits).
Homomorphic Multiplication (c_3, c_4) **mod** N **(once).** Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$. $c = e(c_3, c_4) \cdot h_T^r = g_T^{m_3 \cdot m_4 \mod N} \cdot h_T^{r'} \in G_T$.
Homomorphic Addition (c_5, c_6) **mod** $N \in G_T$**.** Pick a random $r \leftarrow \{0, 1, \ldots, N-1\}$. $c = c_5 \cdot c_6 \cdot h_T^r = g_T^{m_5+m_6 \mod N} \cdot h_T^{r'} \in G_T$.
Decrypt$(\mathcal{SK}, c \in G_T)$**.** Compute c^{p_1} then its discrete log of in base $g_T^{p_1}$.

Implementation. In the Encrypt step of the BGN protocol, a random r is picked in $\{0, 1, \ldots, N-1\}$ with $N = p_1 p_2$ the RSA modulus. Then $u_{1(p_1)}^r$ is computed. The size of r is up to 3072 bits. We used the same curve as in Tab. 6, line with $\log_2 N = 3072$ and $\log_2 p_i = 1536$. We assumed that to compute several pairings on the same curve, we compute each Miller loop, then multiply the outputs and apply a unique final exponentiation. There are four distinct products of two or three pairings in the second protocol.

Table 7. Timings for the BGN protocol over a composite order elliptic curve and its equivalent over a prime order elliptic curve for a security level equivalent to AES-128. We don't consider a discrete log computation because this is not the scope of our paper, see e.g. [4] for efficient DL computation in this particular setting.

Operation	Composite-order E.C. [7, §3]		Prime-order E.C. [10, §5]	
Encrypt or Add	1 exp. in G_1	1300 ms	1 exp. in G_1 and G_2	3.8 ms
Decrypt	$C^{p_1} \in G_1$	645 ms	π_1: 4 exp. in G_1 π_2: 4 exp. in G_2	4.0 ms 11.2 ms
Multiply	1 pairing + 1 exp. in G_T	3364 ms	1 exp. in G_1 and G_2 + 4×(3 pairings)	119.8 ms
Encrypt or Add	1 exp. in G_T	409 ms	1 exp. in G_1 and G_2 + 4×(2 pairings)	87.8 ms
Decrypt (without DL)	$C^{p_1} \in G_T$	204 ms	$\pi_t(C)$ 16 exp. in G_T	108.8 ms

The arithmetic on the composite-order elliptic curve $E(\mathbb{F}_p)$ is more than 3 times slower than in $G_T \subset \mathbb{F}_{p^2}$, this means that the encryptions and exponentiations of decryption in G_T are more efficient. The converse is observed over a prime-order elliptic curve. This protocol over an optimal prime-order elliptic curve is dramatically faster than over a composite-order elliptic curve. More precisely, the exponentiation of the decryption step is 161 times faster in G_1, 57 times faster in G_2 and 2 times faster in G_T over a prime-order elliptic curve than over a composite-order one.

5.2 Application to Hierarchical Identity Based Encryption

In this section, we detail and implement the Hierarchical Identity Based Encryption (HIBE in the following) of Lewko and Waters published at Eurocrypt 2011 [20] and compare it with its translation in the prime-order setting due to Lewko [19].

Lewko-Waters HIBE scheme. We only recall the Setup, KeyGen, Encrypt, Delegate and Decrypt steps. For a complete description of the scheme, see [20].

Setup($\lambda \rightarrow$ PP, MSK). The setup algorithm takes as input the security parameter λ (e.g. see Tab. 2 to select an appropriate λ) and chooses a bilinear group G_1 of order $N = p_1p_2p_3$, where p_1, p_2, p_3 are distinct primes. Let $G_{1(p_i)}$ denote the subgroup of order p_i in G_1. The algorithm then chooses g, u, h, v, w uniformly randomly from $G_{1(p_1)}$, and α uniformly randomly from $\mathbb{Z}_N$. It sets the public parameters as:

$$\text{PP} := \{N, G_1, g, u, h, v, w, e(g,g)^{\alpha}\} \ .$$

The master secret key is α.

KeyGen($(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, MSK, PP) $\rightarrow$ $\mathbf{SK_{\mathcal{I}}}$. The key generation algorithm chooses uniformly at random values $r_1, \ldots, r_j, y_1, \ldots, y_j$ from $\mathbb{Z}_N$. It also chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ subject to the constraint that $\alpha = \lambda_1 + \lambda_2 + \ldots + \lambda_j$. The secret key is created as:

$$K_{i,0} := g^{\lambda_i} w^{y_i},\ K_{i,1} := g^{y_i},\ K_{i,2} := v^{y_i}(u^{\mathcal{I}_i}h)^{r_i},\ K_{i,3} := g^{r_i}\ \forall i \in \{1, \ldots, j\} \ .$$

Encrypt(M, $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, PP), $\rightarrow$ CT. The encryption algorithm chooses s, t_1, ..., t_j uniformly randomly from $\mathbb{Z}_N$. It creates the ciphertext as:

$$C := Me(g,g)^{\alpha s}, C_0 := g^s,$$

$$C_{i,1} := w^s v^{t_i},\ C_{i,2} := g^{t_i},\ C_{i,3} := (u^{\mathcal{I}_i}h)^{t_i}\ \forall i \in \{1, \ldots, j\} \ .$$

Delegate(PP, SK, $\mathcal{I}_{j+1}$) $\rightarrow$ $SK^{'}$. $\mathcal{I}_{j+1}$ denotes the identity of a group under $\mathcal{I}_j$ in the hierarchy. The delegation algorithm takes in a secret key $SK = \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}\ \forall i \in \{1, \ldots, j\}\}$ for $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j)$ and a level $j+1$ identity $\mathcal{I}_{j+1}$. It produces a secret key $SK^{'}$ for $(\mathcal{I}_1, \ldots, \mathcal{I}_{j+1})$ as follows. It chooses $y_1^{'}, \ldots, y_{j+1}^{'}$ and $r_1^{'}, \ldots, r_{j+1}^{'} \in \mathbb{Z}_N$ uniformly at random, $\lambda_1^{'}, \ldots, \lambda_{j+1}^{'} \in \mathbb{Z}_N$ randomly up to the constraint that $\lambda_1^{'} + \ldots + \lambda_{j+1}^{'} = 0$ and computes:

$$K_{i,0}^{'} := K_{i,0} \cdot g^{\lambda_i^{'}} \cdot w^{y_i^{'}},\ K_{i,1}^{'} := K_{i,1} \cdot g^{y_i^{'}},\ K_{i,2}^{'} := K_{i,2} \cdot v^{y_i^{'}}(u^{\mathcal{I}_i}h)^{r_i^{'}},$$

$$K_{i,3}^{'} := K_{i,3} \cdot g^{r_i^{'}}\ \forall i \in \{1, \ldots, j+1\},$$

where $K_{j+1,1}, K_{j+1,2}, K_{j+1,3}$ are defined to be the identity element in G_1.

Decryption(CT, SK) $\rightarrow$ M. The decryption algorithm takes in a secret key $\mathrm{SK} = \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}\ \forall i \in \{1, \ldots, j\}\}$ for $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j)$ and a ciphertext CT encrypted to $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$. Assuming $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ is a prefix of $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$, the message is decrypted as follows. The decryption algorithm computes:

$$B := \prod_{i=1}^{j} \frac{e(C_0, K_{i,0})e(C_{i,2}, K_{i,2})}{e(C_{i,1}, K_{i,1})e(C_{i,3}, K_{i,3})} .$$

The message is then computed as $M = C/B$.

Table 8. Lewko and Waters HIBE scheme over a composite order bilinear group.

Operation	Randomness complexity	Computation	Timing $j = 3$ Tab. 6
Setup	$N = p_1p_2p_3$, 5 elts $\in G_{1(p_1)}$, 1 elt $\in \mathbb{Z}_N$	1 pairing	1.27 s
KeyGen	$3j - 1$ elts in $\mathbb{Z}_N$	$7j$ exp. in G_1	11.55 s
Encrypt	$j + 1$ elts $\in Z_N$	$4 + 4j$ exp. in G_1, 1 exp. in G_T	8.96 s
Delegate $j \rightarrow j + 1$	$3j + 2$ elts in $\mathbb{Z}_N$	$7(j + 1)$ exp. in G_1	15.40 s
Decryption	–	$4j$ pairings	5.08 s

Lewko HIBE translation in prime order bilinear group. We also studied the Lewko HIBE translation in prime order bilinear group. We only consider in Tab. 9 the Setup, Encrypt, KeyGen, Delegate and Decrypt steps written only from practical point of view, with $m = 6$. For a complete description of the scheme with $m = 10$ for the security proof, see [19, §B.3] and [19, §2.2] for notations. Moreover the scheme in [19] is described with a symmetric pairing. We apply the protocol to an asymmetric pairing to improve its practical efficiency. There are two possible approaches. We can set the secret keys in G_1 and the ciphertexts in G_2 to optimise the needs in secured memory which can be quite expensive in constrained devices. Or we can set in G_2 the secrets keys (with double secured memory) and set in G_1 the ciphertexts to improve the bandwidth. We will choose this second option.

Vectors of group elements are considered and denoted $\boldsymbol{v} = (v_1, \ldots, v_m) \in \mathbb{F}_r^m$ (with r the prime subgroup order of an elliptic curve), and for $g_1 \in G_1$ (recall that this is an elliptic curve and not a finite field despite the multiplicative notation),

$$g_1^{\boldsymbol{v}} = (g_1^{v_1}, g_1^{v_2}, \ldots, g_1^{v_m}) \in G_1^m . \tag{3}$$

Moreover, for any $a \in \mathbb{F}_r$ and $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{F}_r^m$, we have:

$$g_1^{a\boldsymbol{v}} = (g_1^{av_1}, g_1^{av_2}, \ldots, g_1^{av_m}),\ g_1^{\boldsymbol{v}+\boldsymbol{w}} = (g_1^{v_1+w_1}, g_1^{v_2+w_2}, \ldots, g_1^{v_m+w_m}) . \tag{4}$$

The corresponding pairing is defined as follows, with e a *one dimensional* bilinear pairing:

$$e_m(g_1^{\boldsymbol{v}}, g_2^{\boldsymbol{w}}) = \prod_{i=1}^{m} e(g_1^{v_i}, g_2^{w_i}) = e(g_1, g_2)^{\boldsymbol{v}\cdot\boldsymbol{w}} \in G_T \subset \mathbb{F}_{p^k}^* \ . \tag{5}$$

The pairing e_m costs m pairings e. More precisely, as e_m is a product of m pairings, it costs m Miller loops then one final exponentiation if we set e to be a (variant of a) Tate pairing.

Setup($\lambda \to$ PP, MSK). The setup algorithm takes in the security parameter λ and chooses a bilinear group G_1 of sufficiently large prime order r and a generator g_1, G_2 of same prime order r and a generator g_2 and G_T of same order r and let $g_T = e(g_1, g_2)$ a generator for G_T. let $e : G_1 \times G_2 \to G_T$ denote the bilinear map. We set $m = 6$. Hence

$$\begin{aligned} e_m = e_6 : G_1^6 \times G_2^6 &\to G_T \\ (g_1^{\boldsymbol{v}}, g_2^{\boldsymbol{w}}) &\mapsto \textstyle\prod_{i=1}^{6} e(g_1^{v_i}, g_2^{w_i}) \end{aligned}$$

The algorithm samples random dual orthonormal bases, $(\mathbb{D}, \mathbb{D}^*) \leftarrow \mathit{Dual}(\mathbb{F}_r^m)$. Let $\boldsymbol{d_1}, \ldots, \boldsymbol{d_6}$ denote the elements of $\mathbb{D}$ and $\boldsymbol{d_1^*}, \ldots, \boldsymbol{d_6^*}$ denote the elements of $\mathbb{D}^*$. They satisfy the property $\boldsymbol{d_i} \cdot \boldsymbol{d_i^*} = \psi \in \mathbb{F}_r^* \forall i$ and $\boldsymbol{d_i} \cdot \boldsymbol{d_j^*} = 0 (\mod r)$ for $i \neq j$. It also chooses random exponents $\alpha_1, \alpha_2, \theta, \sigma, \gamma, \xi \in \mathbb{F}_r$. The public parameters are

$$PP = \left\{ G_1, G_2, G_T, r, e(g_1, g_2)^{\alpha_1 \boldsymbol{d_1}\cdot\boldsymbol{d_1^*}}, e(g_1, g_2)^{\alpha_2 \boldsymbol{d_2}\cdot\boldsymbol{d_2^*}}, g_1^{\boldsymbol{d_1}}, \ldots, g_1^{\boldsymbol{d_6}} \right\},$$

and the master secret key is

$$MSK = \left\{ \alpha_1, \alpha_2, g_2^{\boldsymbol{d_1^*}}, g_2^{\boldsymbol{d_2^*}}, g_2^{\gamma \boldsymbol{d_1^*}}, g_2^{\xi \boldsymbol{d_2^*}}, g_2^{\theta \boldsymbol{d_3^*}}, g_2^{\theta \boldsymbol{d_4^*}}, g_2^{\sigma \boldsymbol{d_5^*}}, g_2^{\sigma \boldsymbol{d_6^*}} \right\} \ .$$

KeyGen($(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, MSK, PP) $\to$ $\mathbf{SK_{\mathcal{I}}}$. The key generation algorithm chooses uniformly at random values $r_1^i, r_2^i \in \mathbb{F}_r$ for $1 \leqslant i \leqslant j$. It also chooses random values $y_1, \ldots, y_j \in \mathbb{F}_r$ and $w_1, \ldots, w_j \in \mathbb{F}_r$ up to the constraint that $y_1 + y_2 + \ldots + y_j = \alpha_1$ and $w_1 + w_2 + \ldots + w_j = \alpha_2$. For each $1 \leqslant i \leqslant j$ it computes $K_i := g_2^{y_i \boldsymbol{d_1^*} + w_i \boldsymbol{d_2^*} + r_1^i \mathcal{I}_i \theta \boldsymbol{d_3^*} - r_1^i \theta \boldsymbol{d_4^*} + r_2^i \mathcal{I}_i \sigma \boldsymbol{d_5^*} - r_2^i \sigma \boldsymbol{d_6^*}} \in G_2$. The secret key is created as:

$$SK_{\mathcal{I}} := \left\{ g_2^{\gamma \boldsymbol{d_1^*}}, g_2^{\xi \boldsymbol{d_2^*}}, g_2^{\theta \boldsymbol{d_3^*}}, g_2^{\theta \boldsymbol{d_4^*}}, g_2^{\sigma \boldsymbol{d_5^*}}, g_2^{\sigma \boldsymbol{d_6^*}}, K_1, \ldots, K_j \in G_2 \right\}.$$

Encrypt(M, $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, PP), $\to$ CT. The encryption algorithm chooses s_1, s_2 and t_1^i, t_2^i for $1 \leqslant i \leqslant j$ uniformly randomly from $\mathbb{F}_r$. It computes

$$C_0 := M e(g_1, g_2)^{\alpha_1 s_1 \boldsymbol{d_1}\cdot\boldsymbol{d_1^*}} e(g_1, g_2)^{\alpha_2 s_2 \boldsymbol{d_2}\cdot\boldsymbol{d_2^*}} \in G_T$$

(note that $e(g_1, g_2)^{\alpha_1 \boldsymbol{d_1}\cdot\boldsymbol{d_1^*}}$ and $e(g_1, g_2)^{\alpha_2 \boldsymbol{d_2}\cdot\boldsymbol{d_2^*}}$ are in PP). It computes also

$$C_i := g_1^{s_1 \boldsymbol{d_1} + s_2 \boldsymbol{d_2} + t_1^i \boldsymbol{d_3} + \mathcal{I}_i t_1^i \boldsymbol{d_4} + t_2^i \boldsymbol{d_5} + \mathcal{I} t_2^i \boldsymbol{d_6}}$$

for $1 \leqslant i \leqslant j$. The ciphertext is $\mathrm{CT} := \{C_0 \in G_T, C_1, \ldots, C_j \in G_1\}$.

Delegate(PP, $SK_{\boldsymbol{\mathcal{I}}}$, $\mathcal{I}_{j+1}$) $\rightarrow$ $SK_{\boldsymbol{\mathcal{I}}|\mathcal{I}_{j+1}}$. The delegation algorithm chooses random values $\omega_1^i, \omega_2^i \in \mathbb{F}_r$ for $1 \leqslant i \leqslant j+1$. It also chooses random values $y_1^{'}, \ldots, y_j^{'} \in \mathbb{F}_r$ and $w_1^{'}, \ldots, w_j^{'} \in \mathbb{F}_r$ up to the constraint that $y_1^{'} + y_2^{'} + \ldots + y_{j+1}^{'} = 0$ and $w_1^{'} + w_2^{'} + \ldots + w_{j+1}^{'} = 0$. The delegation algorithm takes in a secret key $SK_{\boldsymbol{\mathcal{I}}}$ with elements denoted as above. It computes $K_i^{'} := K_i \cdot g_2^{y_i^{'}\gamma \boldsymbol{d_1^*} + w_i^{'}\xi \boldsymbol{d_2^*} + \omega_1^i \mathcal{I}_i \theta \boldsymbol{d_3^*} - \omega_1^i \theta \boldsymbol{d_4^*} + \omega_2^i \mathcal{I}_i \sigma \boldsymbol{d_5^*} - \omega_2^i \sigma \boldsymbol{d_6^*}} \in G_2$ for $1 \leqslant i \leqslant j$ and $K_{j+1} := g_2^{y_{j+1}^{'}\gamma \boldsymbol{d_1^*} + w_{j+1}^{'}\xi \boldsymbol{d_2^*} + \omega_1^{j+1} \mathcal{I}_{j+1} \theta \boldsymbol{d_3^*} - \omega_1^{j+1} \theta \boldsymbol{d_4^*} + \omega_2^{j+1} \mathcal{I}_{j+1} \sigma \boldsymbol{d_5^*} - \omega_2^{j+1} \sigma \boldsymbol{d_6^*}} \in G_2$. $SK_{\boldsymbol{\mathcal{I}}|\mathcal{I}_{j+1}}$ is formed as

$$\left\{ g_2^{\gamma \boldsymbol{d_1^*}}, g_2^{\xi \boldsymbol{d_2^*}}, g_2^{\theta \boldsymbol{d_3^*}}, g_2^{\theta \boldsymbol{d_4^*}}, g_2^{\sigma \boldsymbol{d_5^*}}, g_2^{\sigma \boldsymbol{d_6^*}} (\text{ from } SK_{\boldsymbol{\mathcal{I}}}),\ K_1^{'}, \ldots, K_j^{'}, K_{j+1} \in G_2 \right\} .$$

Decryption(CT, $SK_{\boldsymbol{\mathcal{I}}}$) $\rightarrow$ M. Assuming $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ is a prefix of $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$, the decryption algorithm computes $B := \prod_{i=1}^{j} e_m(C_0, K_i)$. The message is then computed as $M = C_0/B$.

Each step is summarised in Tab. 9. We chose a hierarchy depth of $j = 3$.

Table 9. Lewko HIBE scheme translation over prime order bilinear group.

Operation	Randomness complexity	Computation	Timing Tab. 6 $j = 3, m = 6$
Setup	r, $2m^2$ elts in $\mathbb{F}_r$ for $(\mathbb{D}, \mathbb{D}^*)$, 6 elts $\in \mathbb{F}_r$	1 pairing e, 2 exp. in G_T, m^2 exp. in G_1, $m(m+2)$ exp. in G_2	127 ms
KeyGen	$2j + 2(j-1)$ elts $\in \mathbb{F}_r$	$j \cdot m^2$ exp. in G_2, some mult. in $\mathbb{F}_p$ and G_2	206 ms
Encrypt	$2 + 2j$ elts in $\mathbb{F}_r$	$j \cdot m^2$ exp. in G_1, 2 exp. in G_T, some mult. in $\mathbb{F}_p$	70 ms
Delegate $j \rightarrow j+1$	$2(j+1) + 2j$ elts in $\mathbb{F}_r$	$(j+1)m^2$ exp. in G_2	80 ms
Decryption	–	$j \cdot m$ pairings e	45.0 ms

We can say that this instantiation (Tab. 9) is 10 times more efficient than with a composite-order elliptic curve (Tab 8) for Setup, 56 times for KeyGen, 128 times for Encrypt, 192 times for Delegate and 112 times for Decryption. In other words, the important operations of delegation, encryption and decryption are more than hundred times faster over a prime-order bilinear curve with an asymmetric pairing compared to a composite-order supersingular curve with a symmetric pairing.

6 Conclusion

We studied interesting protocols based on composite-order or prime-order elliptic curves. The composite order must be infeasible to factor. For each elliptic

curve, a discrete logarithm must be impossible to compute (in reasonable time). We justified the sizes of the composite orders when more than two primes are present in the modulus. We analysed the Number Field Sieve complexity and the Elliptic Curve Method to find the size bounds. We then compared the cost of the homomorphic encryption scheme of Boneh, Goh and Nissim over a composite-order and its counterpart over a prime-order pairing-friendly elliptic curve given by Freeman. In the former case, a pairing took 3 s, compared to 13 ms in the latter case. Even with 12 pairings instead of one in the Multiply step of the protocol, the prime-order translation remained 28 times faster. We also compared the unbounded HIBE protocol of Waters and Lewko and its translation given by Lewko. The prime-order setting is between 10 times to 192 times faster than the composite-order setting. Despite useful properties of bilinear composite-order structures to design new protocols, the resulting schemes are not very competitive compared to protocols relying on other assumptions which in particular, need prime-order bilinear structures with asymmetric pairings. Some special protocols need extra properties such as cancelling and projecting pairings. Only composite-order groups or supersingular curves achieve these properties.

We recommend to avoid the needs of composite-order groups whenever possible. Moreover, we did not investigate multi-exponentiation techniques to compute simultaneously several pairings on the same elliptic curve, neither did we use the Frobenius map to decompose exponents when performing exponentiation in $\mathbb{F}_{p^{12}}$. Hence some speed-up are still available for protocols in the prime-order setting.

Acknowledgements. Thanks to Damien Vergnaud for his help in this paper. We thank the reviewers of the ACNS Conference for their useful comments. This work was supported in part by the French ANR-09-VERS-016 BEST Project.

References

1. T. Acar, K. Lauter, M. Naehrig, and D. Shumow. Affine pairings on ARM. In M. Abdalla and T. Lange, editors, *Pairing 2012*, volume 7708 of *LNCS*, pages 203–209. Springer, 2012.
2. D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López. Faster explicit formulas for computing pairings over ordinary curves. In *EUROCRYPT*, volume 6632 of *LNCS*, pages 48–68, 2011.
3. P. S. Barreto and M. Naehrig. Pairing friendly elliptic curves of prime order. In *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, 2006.
4. D. J. Bernstein and T. Lange. Computing small discrete logarithms faster. In *INDOCRYPT*, pages 317–338, 2012.
5. J.-L. Beuchat, J. E. G. Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal ate pairing over Barreto–Naehrig curves. In M. Joye, A. Miyaji, and A. Otsuka, editors, *Pairing*, number 6487 in LNCS, pages 21–39. Springer, 2010.
6. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

7. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *TCC*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.
8. D. Boneh, K. Rubin, and A. Silverberg. Finding composite order ordinary elliptic curves using the Cocks-Pinch method. *Journal of Number Theory*, 131(5):832 – 841, 2011.
9. A. J. Devegili, C. O. hÉigeartaigh, M. Scott, and R. Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471, 2006.
10. D. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.
11. D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.
12. R. Granger and M. Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *LNCS*, pages 209–223. Springer, 2010.
13. M. Herrmann. Improved cryptanalysis of the multi-prime Φ-hiding assumption. In *AFRICACRYPT*, pages 92–99, 2011.
14. G. S. Ian F. Blake and N. P. Smart. *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005.
15. E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In *CRYPTO*, pages 295–313, 2010.
16. N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In *Cryptography and Coding*, volume 3796 of *LNCS*, pages 13–36. Springer, 2005.
17. A. K. Lenstra. Unbelievable security. matching AES security using public key systems. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 67–86, 2001.
18. A. K. Lenstra and H. W. J. Lenstra, editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 1993.
19. A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 318–335. Springer, 2012.
20. A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In *EUROCRYPT*, pages 547–567. Springer, 2011.
21. M. Naehrig, R. Niederhagen, and P. Schwabe. New software speed records for cryptographic pairings. In M. Abdalla and P. S. Barreto, editors, *LATINCRYPT*, volume 6212 of *LNCS*, pages 109–123. Springer, 2010.
22. J. H. Seo. On the (im)possibility of projecting property in prime-order setting. In *ASIACRYPT*, pages 61–79, 2012.
23. F. Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.
24. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009.
25. Y. Zhang, C. Xue, D. Wong, N. Mamoulis, and S. Yiu. Acceleration of composite order bilinear pairing on graphics hardware. In T. Chim and T. Yuen, editors, *Information and Communications Security*, volume 7618 of *LNCS*, pages 341–348. Springer, 2012.
26. C. Zhao, F. Zhang, and D. Xie. Faster computation of self-pairings. *IEEE Transactions on Information Theory*, 58(5):3266–3272, 2012.