Hashing to $\mathbb{G}_2$ on BLS pairing-friendly curves

Alessandro Budroni[1] and Federico Pintore[2]

[1]Department of Informatics, University of Bergen, Norway - alessandro.budroni@uib.no *
[2]Department of Mathematics, University of Trento, Italy - federico.pintore@unitn.it, federico.pintore@gmail.com

May 1, 2018

Abstract

When a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, on an elliptic curve E defined over $\mathbb{F}_q$, is exploited in a cryptographic protocol, there is often the need to hash binary strings into $\mathbb{G}_1$ and $\mathbb{G}_2$. Traditionally, if E admits a twist $\tilde{E}$ of order d, then $\mathbb{G}_1 = E(\mathbb{F}_q) \cap E[r]$, where r is a prime integer, and $\mathbb{G}_2 = \tilde{E}(\mathbb{F}_{q^{k/d}}) \cap \tilde{E}[r]$, where k is the embedding degree of E w.r.t. r. The standard approach for hashing a binary string into $\mathbb{G}_1$ and $\mathbb{G}_2$ is to map it to general points $P \in E(\mathbb{F}_q)$ and $P' \in \tilde{E}(\mathbb{F}_{q^{k/d}})$, and then multiply them by the cofactors $c = \#E(\mathbb{F}_q)/r$ and $c' = \#\tilde{E}(\mathbb{F}_{q^{k/d}})/r$ respectively. Usually, the multiplication by c' is computationally expensive. In order to speed up such a computation, two different methods (by Scott *et al.* and by Fuentes *et al.*) have been proposed. In this poster we consider these two methods for BLS pairing-friendly curves having $k \in \{12, 24, 30, 42, 48\}$, providing efficiency comparisons. When $k = 42, 48$, the Fuentes *et al.* method requires an expensive one-off pre-computation which was infeasible for the computational power at our disposal. In these cases, we theoretically obtain hashing maps that follow Fuentes *et al.* idea.

1 Introduction

In recent years, several cryptographic protocols have been proposed with pairings on elliptic curves as building blocks. For a given finite field $\mathbb{F}_q$ and an elliptic curve E defined over it, a pairing is a bilinear function e that takes as inputs points on $E(\mathbb{F}_q)$ or on $E(\mathbb{F}_{q^k})$ - where $\mathbb{F}_{q^k}$ is an extension field of the base field $\mathbb{F}_q$ - and returns as outputs elements of $(\mathbb{F}_{q^k})^*$. In particular, given a prime r such that $r||\#E(\mathbb{F}_q)$, e is usually of the form:

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

where $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve subgroups of order r defined as:

- $\mathbb{G}_1 = E(\mathbb{F}_q) \cap E[r]$,
- $\mathbb{G}_2 = E[r] \cap \{(x, y) \in E(\mathbb{F}_{q^k}) | (x^q, y^q) = [q](x, y)\}$,

while $\mathbb{G}_T$ is a subgroup of order r of $(\mathbb{F}_{q^k})^*$. With k is denoted the smallest positive integer such that $r \mid q^k - 1$, which is called *embedding degree* of E with respect to r.

*This work was done while Alessandro was an employee at MIRACL Labs, London, England

For pairing-based schemes to be secure, the discrete logarithm problems on both $E(\mathbb{F}_q)$ and $(\mathbb{F}_{q^k})^*$ must be computationally infeasible. Those elliptic curves providing a fixed level of security along with efficiency of computations are called *pairing-friendly elliptic curves*. In recent years, researchers proposed different methods to build *families* of pairing-friendly elliptic curves [17], [3], [4], [9], [13].

New advances on the Number Field Sieve ([2], [14]) for computing discrete logarithms in $\mathbb{G}_T$ decreased the security of some asymmetric parings, including those build on Barreto-Naehrig (BN) curves (see [16] and [1]). In the light of these results, Barreto-Lynn-Scott (BLS) pairing-friendly curves are attracting more interest, also for efficiency reasons.

When pairings on elliptic curves are exploited for identity-based protocols, there is often the need to *hash* binary strings into $\mathbb{G}_1$ and $\mathbb{G}_2$. Hashing to $\mathbb{G}_1$ is relatively easy. In fact, since $\mathbb{G}_1$ is the unique subgroup of order r of $E(\mathbb{F}_q)$, the standard approach is to hash to a general point $P \in E(\mathbb{F}_q)$ and then multiply it by the cofactor $c = \#E(\mathbb{F}_q)/r$. If E admits a twist of degree d that divides k, then $\mathbb{G}_2$ is isomorphic to $\tilde{E}(\mathbb{F}_{q^{k/d}}) \cap \tilde{E}[r]$, where $\tilde{E}$ is a degree d twist of $E/\mathbb{F}_{q^{k/d}}$ [11], and consequently the same approach can be used for hashing into $\mathbb{G}_2$. Nevertheless, the latter requires a multiplication by a large cofactor and hence expensive computations.

1.1 Related Works

In 2009, Scott *et al.* [19] reduced the computational cost of this cofactor multiplication exploiting an efficiently-computable endomorphism $\psi : \tilde{E} \to \tilde{E}$. An improvement of this method was then obtained by Fuentes *et al.* in 2011 [10]. Since pairing-friendly families vary significantly, in order to highlight the benefits of the two methods, families of curves were considered case-by-case in [19] and in [10]. In particular, both papers focus on BN curves with $k = 12$, Freeman curves with $k = 10$ and KSS curves with $k = 8, 18$.

2 Our Contribution

In this poster we report some results from the recent work [7], currently under evaluation by an international journal. We provide an efficiency comparison between the two hashing methods, [19] and [10], for BLS curves with $k = 12, 24$. Such a comparison contrasts with that of a recently-published book [8, Chapter 8], where authors state that for BLS curves with $k = 12, 24$ the most efficient method for hashing into $\mathbb{G}_2$ is the one proposed by Scott *et al.*. Furthermore, we do not know of any publication or source where both Scott *et al.* and Fuentes *et al.* methods have been explicitly applied to BLS curves with $k \in \{30, 42, 48\}$. In this poster that gap is filled for BLS curves having $k = 30$ and also we provide some results about the cases where $k = 42, 48$.

Both Scott *et al.* and Fuentes *et al.* methods require a pre-computation to obtain formulas for a specific family of pairing-friendly curves. Scott *et al.* method needs only polynomial modular arithmetic, while Fuentes *et al.* method goes through the application of the LLL algorithm to a polynomial matrix, in order to obtain a lattice's polynomial $h(z)$ having *small* coefficients. We executed the former computation also for BLS curves having $k = 42, 48$, while the latter computation is prohibitive as the embedding degree k grows. Nevertheless, without LLL algorithm, we provide suitable polynomials $h(z)$ that allow to speed up cofactor multiplications.

In the following table are reported computational costs for hashing into $\mathbb{G}_2$ exploiting the hash maps that we have found. The central column concerns results obtained applying Scott *et al.* method. The last

column contains computational costs obtained following the *Fuentes et al.* method. With 'A' we denote a point addition, with 'D' a point doubling, with 'Z' a scalar multiplication and with 'ψ' an application of the endomorphism ψ. We underline that, in each hashing map, the most significant component is the scalar multiplication, since it computationally dominates other operations. In fact, the algorithms to compute large scalar multiplications require many point additions and doublings. Furthermore, the endomorphism ψ can be efficiently computed.

Curve	Scott et al.	Fuentes et al.
BLS-12	6A 2D 3Z 3ψ	5A 1D 2Z 3ψ
BLS-24	21A 4D 8Z 6ψ	9A 1D 4Z 10ψ
BLS-30	82A 16D 11Z 67ψ	25A 2D 5Z 27ψ
BLS-42	151A 54D 15Z 125ψ	33A 1D 9Z 42ψ
BLS-48	132A 120D 16Z 130ψ	17A 1D 8Z 36ψ

Table 1: Comparison between the computational cost of each hash map.

In all the cases we have examined, the hash map found following the Fuentes *et al.* method turned out to be more efficient than the one found with the Scott *et al.* method. Among the computed hash maps, only those for the cases $k = 12, 24, 30$ are obtained applying rigorously Fuentes *et al.* method. For $k = 12$ we see a 3/2-fold improvement, for $k = 24$ the hash map is twice as fast as that of Scott *et al.* while for $k = 30$ the hash map determines a 11/5-fold improvement. Concerning BLS curves with $k \in \{42, 48\}$, we theoretically propose suitable polynomials $h(z)$ that satisfy conditions of Theorem 1 in [10] ($k = 48$) or that are extremely tight to a polynomial fully satisfying such conditions ($k = 42$). For the case $k = 42$, our proposal leads to a 15/9-fold improvement with respect to the method of Scott *et al.*. For $k = 48$, the introduced hash map is twice as fast as that of Scott *et al.*.

Using the *Apache Milagro Crypto Library* [15] we implemented the hash maps obtained applying Scott *et al.* and Fuentes *et al.* methods on BLS curves with embedding degree $k = 12$. In Table 2 we summarise the timing results of a benchmark test on the two maps.

Processor	Scott et al.	Fuentes et al.
Intel(R) Core(TM) i5-5257U 64-bit - 2.7 GHz	2.83 ms	1.98 ms
Quad-core ARM Cortex A53 64-bit - 1.2 GHz	50.26 ms	35.88 ms

Table 2: *Each value corresponds to the average time (in milliseconds) considered for each hash from a sample of 1000 hashes.*

These experimental results show that the hashing map obtained with the Fuentes *et al.* method is approximately 30% faster than the map obtained with the Scott *et al.* method, as we expected from Table 1.

References

[1] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334, 2017. `http://eprint.iacr.org/2017/334`.

[2] R. Barbulescu, P. Gaudry, and T. Kleinjung. The tower number field sieve. *Advances in Cryptology - ASIACRYPT 2015*, LCNS 9453 (2015):31–55.

[3] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing Elliptic Curves with Prescribed Embedding Degrees. *International Conference on Security in Communication Networks*. Springer Berlin Heidelberg, 2002.

[4] P. S. L. M. Barreto and M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. *International Workshop on Selected Areas in Cryptography*, pages 319–331. Springer Berlin Heidelberg, 2005.

[5] A. Budroni and F. Pintore. Efficient hash maps to $\mathbb{G}_2$ on BLS curves. *IACR Cryptology ePrint Archive 2017, (2017): 419*, 2017.

[6] N. El Mrabet and M. Joye. *Guide to Pairing-Based Cryptography*. Cryptography and Network Security. Chapman and Hall/CRC, 1st edition, 2017.

[7] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. *Algorithmic Number Theory Symposium*, pages 452–465. Springer Berlin Heidelberg, 2006.

[8] L. Fuentes-Castaneda, E. Knapp, and F. Rodriguez-Henriquez. Faster hashing to $\mathbb{G}_2$. *International Workshop on Selected Areas in Cryptography*, pages 412–430. Springer Berlin Heidelberg, 2011.

[9] F. Hess, N. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, Oct. 2006.

[10] E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field. *International Conference on Pairing-Based Cryptography*, pages 126–135. Springer Berlin Heidelberg, 2008.

[11] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for medium prime case. *Annual Cryptology Conference*, LCNS 9814 (2016):543–571. Springer Berlin Heidelberg, 2016.

[12] M. Labs. Apache Milagro Crypto Library (AMCL). `https://github.com/milagro-crypto/milagro-crypto-c`.

[13] A. Menezes, P. Sarkar, and S. Singh. Challenges with Assessing the Impact of NFS Advances on the Security of Pairing-based Cryptography. *Proceedings of Mycrypt.*, 2016.

[14] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals,*, E84 A, no. 5, 1234-1243, May 2001.

[15] M. Scott, N. Benger, M. Charlemagne, L. J. D. Perez, and E. J. Kachisa. Fast hashing to G2 on pairing friendly Curves. *International Conference on Pairing-Based Cryptography*, pages 102–113. Springer Berlin Heidelberg, 2009.