SCIS 2012 The 29th Symposium on Cryptography and Information Security
Kanazawa, Japan, Jan. 30 - Feb. 2, 2012
The Institute of Electronics, Information and Communication Engineers

A Note on Hashing to BN Curves

Mehdi Tibouchi*

Abstract— A number of recent works have considered the problem of constructing constant-time hash functions to various families of elliptic curves over finite fields. In the relevant literature, it has been occasionally asserted (including by the author of this note) that constant-time hashing to certain special elliptic curves, namely those of j-invariant 0, was an open problem. That is actually incorrect, as the problem was previously solved in full generality by Shallue and van de Woestijne, back in 2006. The purpose of this note is to introduce the problem of hashing to elliptic curves, and make Shallue and van de Woestijne's solution explicit as well as suggest possible optimizations in the most important of the aforementioned special cases, that of Barreto-Naehrig pairing-friendly elliptic curves.

Keywords: Elliptic curve cryptography, hash function.

1 Introduction

This note first introduces the general problem of hashing to elliptic curves in constant time and discusses on several examples why this problem is of interest to elliptic curve-based cryptographic protocols, especially pairing-based ones. It then turns to the specific problem of hashing to Barreto-Naehrig (BN) curves, one of the most important families of pairing-friendly curves. It has been asserted in the literature that this problem was open, but it is in fact covered by the general construction proposed by Shallue and van de Woestijne back in 2006. That general construction can be rather inefficient in most cases, but the special form of BN curves allows a number of optimizations that make the encoding almost, but not quite, competitive with constructions like Icart's, that apply to other families of curves.

1.1 Background

Many elliptic curve-based cryptographic protocols require hashing to the elliptic curve group $\mathbb{G}$: they involve one or more hash functions $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$ mapping arbitrary values to points on the elliptic curve.

For example, in the Boneh-Franklin identity-based encryption scheme [4], the public key for identity $\mathrm{id} \in \{0,1\}^*$ is a point $\mathbf{Q}_{\mathrm{id}} = \mathfrak{H}(\mathrm{id})$ on the curve. This is also the case in many other pairing-based cryptosystems including IBE and HIBE schemes [1, 16, 17], signature and identity-based signature schemes [3, 5, 6, 11, 31] and identity-based signcryption schemes [8, 22].

Hashing into elliptic curves is also required for some passwords-based authentication protocols such as the SPEKE [19] and PAK [9] protocols, as well as various signature schemes based on the hardness of the discrete logarithm problem, like [12], when they are instantiated over elliptic curves.

In all of those cases, the hash functions are modeled as random oracles in security proofs. However, it is not clear how such a hash function can be instantiated in practice. Indeed, random oracles to groups like $\mathbb{Z}_p^*$ can be easily constructed from random oracles to fixed-length bit strings, for which conventional cryptographic hash functions usually provide acceptable substitutes. On the other hand, constructing random oracles to an elliptic curves even from random oracles to bit strings appears difficult in general, and some of the more obvious instantiations actually break security completely.

1.2 Outline

We first present in §2 a naive construction of an elliptic curve-valued hash function and show on a concrete example how this naive construction breaks security completely. We then introduce in §3 a better solution, the so-called "try-and-increment" hashing, that has satisfactory black-box security properties (it preserves random oracle proofs of security). However, it has the drawback of not running in constant time, which, as we shall see, can be a security concern for physical implementations. We then discuss in §4 another strategy for constructing elliptic curve-valued hash functions, based on simpler building blocks called encodings. Finally, in §5, we turn to the problem of constructing encodings to BN elliptic curves in particular.

2 The trivial encoding

To gain a sense of why the construction of hash functions to elliptic curves requires some care, we first show how a naive construction can completely break the security of a protocol that uses it.

* NTT Information Sharing Platform Laboratories, 3–9–11 Midori-cho, Musashino-shi, Tokyo 180–8585, Japan. tibouchi.mehdi@lab.ntt.co.jp

2.1 A naive construction

We would like to construct a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$ to an elliptic curve group $\mathbb{G}$, which we can assume is cyclic of order N and generated by a given point G. The simplest, most naive way to do so is probably to start from an integer-valued hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_N$ (for which reasonable instantiations are easy to come by) and to define $\mathfrak{H}$ as:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathsf{G}. \tag{1}$$

This is, however, a bad idea on multiple levels.

On the one hand, it is easy to see why this will typically break security *proofs* in the random oracle model. Indeed, at some point in a random oracle model security reduction, the simulator will typically want to "program" the random oracle by setting some of its outputs to specific values. In this case, it will want to set the value $\mathfrak{H}(m)$ for some input m to a certain elliptic curve point P. However, if $\mathfrak{H}$ is defined as in (1), the simulator should actually program the integer-valued random oracle $\mathfrak{h}$ to satisfy $[\mathfrak{h}(m)] \cdot \mathsf{G} = \mathsf{P}$. In other words, it should set $\mathfrak{h}(m)$ to the discrete logarithm of P with respect to G. But this discrete logarithm isn't usually known to the simulator, and it cannot be computed efficiently: therefore, the security reduction breaks down.

On the other hand, it may not be immediately clear how this problem translates into an actual security weakness for a protocol using the hash function $\mathfrak{H}$: one could think that it is mostly an artifact of the security proof. Nevertheless, a construction like (1) makes it possible for an adversary to compute the discrete logarithm of $\mathfrak{H}(m)$ whenever m is known, which certainly feels uncomfortable from a security standpoint. We demonstrate below that this discomfort is entirely warranted, by showing that the Boneh-Lynn-Shacham signature scheme [6]—certainly the best-known signature scheme that involves hashing to elliptic curves—becomes completely insecure if the hash function involved is instantiated as in (1).

2.2 BLS signatures

Proposed by Boneh, Lynn and Shacham in 2001 [6], the BLS signature scheme remains the efficient scheme which achieves the shortest signature length to this day: about 160 bits at the 80-bit security level.

It is also quite simple to describe. The public parameters are a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric non degenerate bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (extending the construction to an asymmetric pairing is easy) and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$. A generator G of the group is also fixed.

In practical instances, $\mathbb{G}$ is a prime order subgroup in the group of rational points of a supersingular elliptic curve over a finite field and e is the modified Weil pairing. Therefore, we denote the group law of $\mathbb{G}$ additively, and that of $\mathbb{G}_T$ multiplicatively. The signature scheme is then as described in Figure 1. Boneh,

- $\mathsf{KeyGen}()$: Pick $x \xleftarrow{\$} \mathbb{Z}_p$ as the private key, and $\mathsf{P} \leftarrow [x] \cdot \mathsf{G}$ as the public key.
- $\mathsf{Sign}(m, x)$: Compute the signature as $\mathsf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- $\mathsf{Verify}(m, \mathsf{S}, \mathsf{P})$: accept if and only if $e(\mathfrak{H}(m), \mathsf{P}) = e(\mathsf{S}, \mathsf{G})$.

Figure 1: The BLS signature scheme.

Lynn and Shacham prove that if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$, then this scheme is secure (in the usual sense of existential unforgeability under chosen message attacks) when $\mathfrak{H}$ is modeled as a random oracle.

Now consider the case when $\mathfrak{H}$ is instantiated as in (1). Then, the signature on a message m can be written as:

$$\mathsf{S} = [x] \cdot \mathfrak{H}(m) = \big[x\mathfrak{h}(m)\big] \cdot \mathsf{G} = [\mathfrak{h}(m)] \cdot \mathsf{P}$$

and hence, one can forge a signature on any message using only publicly available data! There is no security left at all when using the trivial hash function construction.

A slightly less naive variant of the trivial construction consists in defining $\mathfrak{H}$ as:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathsf{Q} \tag{2}$$

where Q is an auxiliary public point distinct from the generator G and whose discrete logarithm α with respect to G is not published. Using this alternate construction for $\mathfrak{H}$ thwarts the key-only attack described above against BLS signatures. However, the scheme remains far from secure. Indeed, the signature on a message m can be written as:

$$\mathsf{S} = \big[x\mathfrak{h}(m)\big] \cdot \mathsf{Q} = \big[\alpha x\mathfrak{h}(m)\big] \cdot \mathsf{G} = [\mathfrak{h}(m)] \cdot \alpha\mathsf{P}.$$

Now suppose an attacker knows a valid signature S_0 on some message m_0. Then the signature S on an arbitrary m is simply:

$$\mathsf{S} = \left[\frac{\mathfrak{h}(m)}{\mathfrak{h}(m_0)}\right] \cdot [\mathfrak{h}(m_0)] \cdot [\alpha]\mathsf{P} = \left[\frac{\mathfrak{h}(m)}{\mathfrak{h}(m_0)}\right] \cdot \mathsf{S}_0$$

where the division is computed in $\mathbb{Z}_p$. Thus, even with this slightly less naive construction, knowing a single valid signature is enough to produce forgeries on arbitrary messages: again, a complete security break down.

3 Try-and-increment

A classical construction of a hash function to elliptic curves which does work (and one variant of which is suggested by Boneh, Lynn and Shacham in the original short signatures paper [6]) is the so-called "try-and-increment" algorithm. We present this algorithm here, as well as some of the limitations that explain why different constructions may be preferable.

3.1 The try-and-increment algorithm

Consider an elliptic curve E over a finite field $\mathbb{F}_q$ of odd characteristic. It admits a Weierstrass equation of the form:

$$E\colon y^2 = x^3 + ax^2 + bx + c \tag{3}$$

for some $a, b, c \in \mathbb{F}_q$. A probabilistic way of constructing points on $E(\mathbb{F}_q)$ is then to pick a random $x \in \mathbb{F}_q$, check whether $t = x^3 + ax^2 + bx + c$ is a square in $\mathbb{F}_q$, and if so, let $y = \pm\sqrt{t}$ and return (x, y). If t is not a square, then x is not the abscissa of a point on the curve: then, one can pick another x and try again, and if so, let $y = \pm\sqrt{t}$ and return (x, y). If t is not a square, then x is not the abscissa of a point on the curve: then, one can pick another x and try again. It is an easy consequence of the Hasse bound on the number of points on $E(\mathbb{F}_q)$ that the success probability of a single try is very close to $1/2$.

Now this point construction algorithm can be turned into a hash function based on an $\mathbb{F}_q$-valued random oracle $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$. To hash a message m, the idea is to pick the x-coordinate as, essentially, $\mathfrak{h}(m)$ (which amounts to picking it at random once) and carry out the point construction above. However, since one should also be able to retry in case the first x-coordinate that is tried out is not the abscissa of an actual curve point, we rather let $x \leftarrow \mathfrak{h}(c\|m)$, where c is a fixed length counter initially set to 0 and incremented in case of a failure. Since there is a choice of sign to make when taking the square root of $t = x^3 + ax^2 + bx + c$, we also modify $\mathfrak{h}$ to output an extra bit for that purpose: $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$. This is the try-and-increment algorithm, described more precisely in Algorithm 1 (and called MapToGroup in [6]). The failure probability after up to k iterations is about 2^{-k} by the previous computations, so choosing the length of the counter c to be large enough for up to $k \approx 128$ iterations, say, is enough to ensure that the algorithm succeeds except with negligible probability.

Algorithm 1 The try-and-increment algorithm.

```
1:  procedure TryAndIncrementHash($m$)   ▷ hash to $E\colon y^2 = x^3 + ax^2 + bx + c$
2:      $c \leftarrow 0$                          ▷ $c$ is a $\lceil \log_2 k \rceil$-bit bit string
3:      $(x, b) \leftarrow \mathfrak{h}(c\|m)$               ▷ $\mathfrak{h}$ is a RO to $\mathbb{F}_q \times \{0,1\}$
4:      $t \leftarrow x^3 + ax^2 + bx + c$
5:      if $t$ is a square in $\mathbb{F}_q$ then
6:          $y \leftarrow (-1)^b \cdot \sqrt{t}$        ▷ define $\sqrt{\cdot}$ e.g. as the smaller square root wrt some total order on $\mathbb{F}_q$
7:          return $(x, y)$
8:      else
9:          $c \leftarrow c + 1$
10:         if $c < k$ then
11:             goto step 3
12:         end if
13:     end if
14:     return $\perp$
15: end procedure
```

Boneh, Lynn and Shacham prove that this construction can replace the random oracle $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_q)$ in BLS signatures without compromising security. In fact, it is not hard to see that it is *indifferentiable* from such a random oracle, in the sense of Maurer, Renner and Holenstein [23]: this ensures that this construction can be plugged in many protocols[1] requiring a random oracle $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_q)$ while preserving random oracle security proofs, as discussed in [10].

Nevertheless, there are various reasons why Algorithm 1 is not a completely satisfying construction for hash functions to elliptic curves. There is arguably a certain lack of mathematical elegance in the underlying idea of picking x-coordinates at random until a correct one is found, especially as the length of the counter, and hence the maximum number of trials, has to be fixed (to prevent collisions). More importantly, this may have adverse consequences for the security of physical devices implementing a protocol using this construction: for example, since the number of iterations in the algorithm depends on the input m, an adversary can obtain information on m by measuring the running time or the power consumption of a physical implementation.

3.2 Timing attacks on key agreement protocols

A concrete situation in which this varying running time can be a serious issue is the case of embedded devices (especially e-passports) implementing an elliptic curve-based Password-Authenticated Key Exchange (PAKE) protocol.

PAKE is a method for two parties sharing a common low-entropy secret (such as a four-digit PIN, or a self-picked alphabetic password) to derive a high-entropy session key for secure communication in an authenticated way. One of the main security requirements is, informally, that an attacker should not be able to gain any information about the password, except through a brute force online dictionary attack (i.e. impersonating one of the parties in the protocol and attempting to authenticate with each password, one password at a time), which can be prevented in practice by latency, smart card blocking and other operational measures. In particular, a PAKE protocol should be considered broken if a *passive* adversary can learn any information about the password.

Now consider the PAKE protocol described in Figure 2, which is essentially Jablon's Simple Password-base Exponential Key Exchange (SPEKE) [19] implemented over an elliptic curve, except with a random salt as suggested in [20]. The public parameters are an elliptic curve group $\mathbb{G}$ of prime order p and a hash function $\mathfrak{H}\colon \{0,1\}^* \to \mathbb{G}$. The two parties share a common password π, and derive a high-entropy $\mathsf{K} \in \mathbb{G}$ using Diffie-Hellman key agreement in $\mathbb{G}$ but with a variable generator $\mathsf{G} \in \mathbb{G}$ computed by hashing the password.

[1] Not necessarily *all* protocols, as conventional wisdom would have it until recently, but at least all protocols with single-stage security games, as clarified by Ristenpart, Shacham and Shrimpton [24].

$$\begin{array}{rcl}
\text{Alice (Passport)} & & \text{Bob (Reader)} \\
\hline
 & \xleftarrow{s} & s \overset{\$}{\leftarrow} \{0,1\}^k \\
\mathbf{G} \leftarrow \mathfrak{H}(s\|\pi) & & \mathbf{G} \leftarrow \mathfrak{H}(s\|\pi) \\
r_A \overset{\$}{\leftarrow} \mathbb{Z}_p & & r_B \overset{\$}{\leftarrow} \mathbb{Z}_p \\
\mathbf{A} \leftarrow [r_A]\cdot\mathbf{G} & \xrightarrow{\mathbf{A}} & \\
 & \xleftarrow{\mathbf{B}} & \mathbf{B} \leftarrow [r_B]\cdot\mathbf{G} \\
\mathbf{K} \leftarrow [r_A]\cdot\mathbf{B} & & \mathbf{K} \leftarrow [r_B]\cdot\mathbf{A}
\end{array}$$

Figure 2: A randomized variant of SPEKE.

But if the hash function $\mathfrak{H}$ is instantiated by the try-and-increment construction and an eavesdropper is able to measure the running time of one of the parties, he will find different running times or different power traces depending on how many trials it takes to find a suitable x-coordinate in the computation of $\mathfrak{H}(s\|\pi)$. Since it takes a single iteration with probability close to $1/2$, an execution of the protocol provides at least one bit of information about π to the adversary (and about $-\sum_{k\geqslant 1} 2^{-k}\log_2(2^{-k}) = 2$ bits on average).

This leads to a so-called "partition attack", conceptually similar to those described by Boyd et al. in [7]: the adversary can count the number of iterations needed to compute $\mathfrak{H}(s\|\pi_0)$ for each password π_0 in the password dictionary, keeping only the π_0's for which this number of iterations matches the side-channel measurement. This reduces the search space by a factor of at least 2 (and more typically 4) for each execution of the protocol, as the running times for different values of s are independent. As a result, the eavesdropper can typically reduce his search space to a single password after at most a few dozen executions of the protocol!

A rather inefficient countermeasure that can be considered is to run all k iterations of the try-and-increment algorithm every time. However, even that is probably insufficient to thwart the attack: indeed, the usual algorithm (using quadratic reciprocity) for testing whether an element of $\mathbb{F}_q$ is a square, as is done in Step 5 of Algorithm 1, also has different running times depending on its input. This can provide information to the adversary as well, unless this part is somehow tweaked to run in constant time, which seems difficult to do short of computing the quadratic character with an exponentiation and making the algorithm prohibitively slow with k exponentiations every time. In principle, padding the quadratic reciprocity-based algorithm with dummy operations might provide a less computationally expensive solution, but implementing such a countermeasure securely seems quite daunting. A construction that naturally runs in constant time would certainly be preferable.

4 Elliptic curve encodings

4.1 Main idea

A natural way to construct a constant-time hash function to an elliptic curve E would be to use, as a building block, a suitable function $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ that can be efficiently computed in constant time[2]. Then, combining f with a hash function $\mathfrak{h}\colon \{0,1\}^* \to \mathbb{F}_q$, we can hope to obtain a well-behaved hash function to $E(\mathbb{F}_q)$.

Of course, not all such functions f are appropriate: for example, when $q = p$ is prime, the trivial encoding described in §2 is essentially of that form, with $f\colon u \mapsto [\hat{u}]\cdot\mathbf{G}$ (and $u \mapsto \hat{u}$ some lifting of $\mathbb{F}_p$ to $\mathbb{Z}$).

On the other hand, if f is a bijection between $\mathbb{F}_q$ and $E(\mathbb{F}_q)$ (possibly up to a bounded number of exceptional points) whose inverse is also efficiently computable, then the following construction:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}(m)\big) \tag{4}$$

is well-behaved, in the sense that if $\mathfrak{h}$ is modeled as a random oracle to $\mathbb{F}_q$, then $\mathfrak{H}$ can replace a random oracle to $E(\mathbb{F}_q)$ in any protocol while preserving proofs of security in the random oracle model. Indeed, contrary to what happens in the case of the trivial encoding (where programming the random oracle would require computing discrete logarithm), a simulator can easily choose a value $\mathfrak{H}(m_0) = \mathbf{P}_0$ by setting $\mathfrak{h}(m_0) = f^{-1}(\mathbf{P}_0)$. In fact, such a construction is, again, indifferentiable from a random oracle to $E(\mathbb{F}_q)$.

More generally, we will be considering cases where f is not necessarily an efficiently invertible bijection but only a so-called *samplable* mapping, in the sense that for each $\mathbf{P} \in E(\mathbb{F}_q)$, one can compute a random element of $f^{-1}(\mathbf{P})$ in probabilistic polynomial time.

4.2 A simple example

It was actually one of the first papers requiring hashing to elliptic curves, namely Boneh and Franklin's construction [4] of identity-based encryption from the Weil pairing, that introduced the first practical example of a hash function of the form (4). Boneh and Franklin used elliptic curves of a very special form:

$$E\colon y^2 = x^3 + b \tag{5}$$

over a field $\mathbb{F}_q$ such that $q \equiv 2 \pmod 3$. In $\mathbb{F}_q$, $u \mapsto u^3$ is clearly a bijection, and thus each element has a unique cube root. This makes it possible, following Boneh and Franklin, to define a function f as:

$$\begin{aligned} f\colon \mathbb{F}_q &\to E(\mathbb{F}_q) \\ u &\mapsto \Big(\big(u^2 - b\big)^{1/3}, u\Big). \end{aligned} \tag{6}$$

In other words, instead of picking the x-coordinate and try to deduce the y-coordinate by taking a square root

[2] It is probably superfluous to point out that, by constant time, we mean "whose running time does not depend on the input" (once the choice of parameters like E and $\mathbb{F}_q$ is fixed), and not $O(1)$ time in the sense of complexity theory.

(which may not exist) as before, we first choose the y-coordinate and deduce the x-coordinate by taking a cube root (which always exists).

Obviously, the function f is a bijection from $\mathbb{F}_q$ to all the finite points of $E(\mathbb{F}_q)$. In particular, this implies that $\#E(\mathbb{F}_q) = 1 + \#\mathbb{F}_q = q + 1$; thus, E is supersingular (and hence comes with a computable symmetric pairing). This also means that f satisfies the conditions mentioned in the previous section; therefore, construction (4) can replace the random oracle $\mathfrak{H}$ required by the Boneh-Franklin IBE scheme, or any other protocol proved secure in the random oracle model. And it can also easily be computed in constant time: it suffices to compute the cube root as an exponentiation to a fixed power α such that $3\alpha \equiv 1 \pmod{q-1}$.

Note that in fact, the group $\mathbb{G}$ considered by Boneh and Franklin isn't $E(\mathbb{F}_q)$ itself, but a subgroup $\mathbb{G} \subset E(\mathbb{F}_q)$ of prime order. More precisely, the cardinality q of the base field is chosen of the form $6p - 1$ for some prime $p \neq 2, 3$. Then $E(\mathbb{F}_q)$ has a unique subgroup $\mathbb{G}$ of order p (and index 6), which is the group actually used in the scheme. Hashing to $\mathbb{G}$ rather than $E(\mathbb{F}_q)$ is then easy:

$$\mathfrak{H}(m) = f'\big(\mathfrak{h}(m)\big) \quad \text{where} \quad f'(u) = [6]\cdot f(u). \tag{7}$$

The encoding f' defined in that way isn't injective but it is samplable: indeed, to compute a random preimage of some point $\mathbf{P} \in \mathbb{G}$, we can simply compute the six points $\mathbf{Q}_i$ such that $[6]\cdot\mathbf{Q}_i = \mathbf{P}$, and return $f^{-1}(\mathbf{Q}_i)$ for a random index i. Using that observation, Boneh and Franklin prove that construction (7) can replace the random oracle to $\mathbb{G}$ in their IBE scheme.

4.3 Beyond supersingular curves

The previous example suggests that a sensible first step towards constructing well-behaved constant-time hash functions to elliptic curves is to first obtain mappings $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ that are computable in deterministic polynomial time and samplable, and admit constant-time implementations. We will refer to such mappings as *encoding functions* or simply *encodings*. Note that despite what the name might suggest, there is no assumption of injectivity for those mappings.

It turns out that constructing encodings to elliptic curves beyond special cases such as (6) is far from an easy task. In fact, Schoof mentioned the presumably easier problem of constructing a *single* non-identity point on a general elliptic curve over a finite field (the Hasse bound ensures that there is always such a point over fields of cardinality at least 5) as open in his 1985 paper on point counting [27], and little progress was made on this problem before the 2000s.

The first significant result in that direction was obtained by Schinzel and Skałba in 2004 [26]. They exhibited one non-identity point on the elliptic curve (5) without any assumption on the cardinality of the base field. A part of their result is as follows.

Theorem 1 ([26, Th. 1])**.** *Let $\mathbb{F}_q$ be a finite field of characteristic at least 5 and b an element of $\mathbb{F}_q$ such that $b^3 + 72^3 \neq 0$. Further set:*

$$y_1 = -2^{-9}3^{-5}b^3 + 2^{-6}3^{-3}b^2 - 2^{-3}b - 3$$
$$y_2 = 2^{-8}3^{-6}b^3 - 2^{-5}3^{-3}b^2 + 2^{-2}3^{-1}b + 2$$
$$y_3 = \frac{b^6 - 2^5 3^2 b^5 + 2^6 3^6 b^4 - 2^{10}3^6 5 b^3}{2^8 3^5 (b+72)^3} + \frac{2^{12}3^8 5 b^2 - 2^{16}3^{11} b + 2^{18}3^{12}}{2^8 3^5 (b+72)^3}$$
$$y_4 = \frac{b^9 - 2^3 3^2 7 b^8 + 2^9 3^5 b^7 - 2^{13}3^7 b^6}{2^{10}3^5(b^2 - 72b + 72^2)^3} + \frac{2^{13}3^8 29 b^5 - 2^{17}3^{12} b^4 + 2^{19}3^{13} 7 b^3}{2^{10}3^5(b^2 - 72b + 72^2)^3} + \frac{2^{22}3^{14} b^2 + 2^{24}3^{17} b + 2^{27}3^{18}}{2^{10}3^5(b^2 - 72b + 72^2)^3}.$$

Then for at least one $j = 1, 2, 3, 4$, $y_j^2 - b$ is a cube x^3 in $\mathbb{F}_q$, and hence (x, y_j) is a non-identity rational point on the elliptic curve $y^2 = x^3 + b$.

This rather contrived result only shows how to construct a single non trivial $\mathbb{F}_q$-point on the special curve $E\colon y^2 = x^3 + b$ in deterministic polynomial time.

Further research in recent years by Skałba and others led to the construction of more or less practical encoding functions to all elliptic curves in recent years, but almost all proposed results exclude (either explicitly or implicitly) the precise case of $E\colon y^2 = x^3 + b$ over fields $\mathbb{F}_q$ with $q \equiv 1 \pmod 3$, of which BN curves are the most important examples. This has led some researchers (including the author) to believe that the problem of hashing to BN curves was open. We briefly review those results in the next section, and point to the single exception which does allow to encode to BN curves, rather efficiently at that.

5 Encoding to BN curves

5.1 BN curves

BN curves are a family of pairing-friendly elliptic curves over large prime fields, introduced in 2005 by Barreto and Naehrig [2]. They are one of the preferred families for implementing asymmetric pairings nowadays, as they achieve essentially optimal parameters for obtaining bilinear groups at the 128-bit security level. Indeed, BN curves are of prime order (in particular they satisfy $\rho = 1$) and embedding degree $k = 12$; thus, the pairing on a BN curve over a 256-bit prime field $\mathbb{F}_p$ takes its values in the field $\mathbb{F}_{p^{12}}$ of size $256 \times 12 = 3072$. Then, solving the discrete logarithm problem both in the group of points of the curve and in $\mathbb{F}_{p^k}^\times$ takes time about 2^{128} as required.

The details of the construction of BN curves, based on the CM method, is not really relevant for our purposes. Suffice it to say that Barreto and Naehrig's algorithm outputs an elliptic curve of the form:

$$E\colon y^2 = x^3 + b \tag{8}$$

over a field $\mathbb{F}_p$ with $p \equiv 1 \pmod 3$ (for convenience, they suggest to pick a p satisfying, more precisely, $p \equiv$

31 (mod 36)), such that $\#E(\mathbb{F}_p)$ is prime, together with the generator $\mathbf{G} = (1, \sqrt{b+1} \bmod p) \in E(\mathbb{F}_p)$. Moreover, b is typically a very small integer (the smallest > 0 such that $b+1$ is a quadratic residue mod p).

5.2 A short review of elliptic curves encodings

Although the literature on encoding to elliptic curves has become quite vast in recent years, we can broadly divide the proposed constructions into two families: Icart-type encodings on the one hand, and Skałba-type encodings (which are referred to as "SWU-type" in [14]) in the other.

Icart-type encodings include Icart's own encoding [18] and the variants proposed by Farashahi [15], Kammerer et al. [21] and Couveignes and Kammerer [13]. They derive from the construction proposed by Icart in 2009, who tried to extend the original idea of Boneh and Franklin described in §4.2 to the case of ordinary curves. Like Boneh and Franklin's encoding, they crucially rely on the possibility of taking arbitrary cube roots in the base field $\mathbb{F}_p$, so that p has to satisfy $p \equiv 2 \pmod 3$. Since this is not verified for BN curves, none of these constructions apply to them, and it seems very unlikely that a further variant could cover the BN case.

Skałba-type encodings are obtained in a very different way, by constructing rational curves on certain higher-dimensional varieties associated to the elliptic curves of interest and deducing a map to elliptic curve points. Skałba's own result, published in 2005, can be stated as follows.

Theorem 2 ([29, Th. 1]). *Let $\mathbb{F}_q$ be a finite field of characteristic at least 5, and $g(x) = x^3 + ax + b \in \mathbb{F}_q[x]$ a polynomial over $\mathbb{F}_q$ with $a \neq 0$. Then there exists non constant rational functions $X_1, X_2, X_3, U \in \mathbb{F}_q(x)$ such that the following identity holds in $\mathbb{F}_q(t)$:*

$$g\big(X_1(t^2)\big)g\big(X_2(t^2)\big)g\big(X_3(t^2)\big) = U(t)^2. \tag{9}$$

Identity (9), easily gives rise to an encoding function to the elliptic curve:

$$E\colon y^2 = x^3 + ax + b.$$

For any value $u \in \mathbb{F}_q$ such that u^2 is not a pole of the X_i's, we see that the product

$$g\big(X_1(u^2)\big)g\big(X_2(u^2)\big)g\big(X_3(u^2)\big)$$

is a square in $\mathbb{F}_q$, which implies that at least one of the three values $g\big(X_i(u^2)\big)$ is a quadratic residue. If i is the smallest index for which that is the case, we can then set:

$$(x, y) = \Big(X_i(u^2), \sqrt{g\big(X_i(u^2)\big)}\Big)$$

and (x, y) is then a non trivial point in $E(\mathbb{F}_q)$.

This defines an encoding to any elliptic curve over finite fields of characteristic $\neq 2, 3$, except those of j-invariant 0, which inconveniently include BN curves.

Encoding functions based on largely similar principles were later proposed by Ulas [30], Brier et al. [10] and Sato and Hakuta [25], but they all require $a \neq 0$, and hence exclude BN curves again.

The only exception that the author is aware of is the general construction given by Shallue and van de Woestijne in 2006 [28], which has a lot in common with Skałba's approach, but is the only one that lifts the restriction on the j-invariant being nonzero.

5.3 The SW encoding to BN curves

Consider the general Weierstrass equation for an elliptic curve in odd characteristic (possibly including 3):

$$E\colon y^2 = x^3 + a_2x^2 + a_4x + a_6.$$

Let further $g(x) = x^3 + a_2x^2 + a_4x + a_6 \in \mathbb{F}_q[x]$. It is possible to construct an encoding function to $E(\mathbb{F}_q)$ like before from a rational curve on the three-dimensional variety:

$$V\colon y^2 = g(x_1)g(x_2)g(x_3)$$

(which, geometrically, is the quotient of $E \times E \times E$ by $(\mathbb{Z}/2\mathbb{Z})^2$, where each non trivial element acts by $[-1]$ on two components and by the identity on the third one). Indeed, if $\phi\colon \mathbb{A}^1 \to V$, $t \mapsto \big(x_1(t), x_2(t), x_3(t), y(t)\big)$ is such a rational curve, then for any $u \in \mathbb{F}_q$ that is not a pole of ϕ, at least one of $g\big(x_i(u)\big)$ for $i = 1, 2, 3$ is a quadratic residue, and we obtain a corresponding point on $E(\mathbb{F}_q)$ like before.

Then, Shallue and van de Woestijne show how to construct such a rational curve ϕ (and in fact a large number of them). They first obtain an explicit rational map $\psi : S \to V$, where S is the surface of equation:

$$S\colon y^2 \cdot \big(u^2 + uv + v^2 + a_2(u + v) + a_4\big) = -g(u)$$

which can also be written, by completing the square with respect to v, as:

$$\big[y(v + \frac{1}{2}u + \frac{1}{2}a)\big]^2 + \big[\frac{3}{4}u^2 + \frac{1}{2}a_2u + a_4 - \frac{1}{4}a_2^2\big]y^2 = -g(u).$$

Now observe that for any fixed $u \in \mathbb{F}_q$, the previous equation defines a curve of genus 0 in the (v, y)-plane. More precisely, it can be written as:

$$z^2 + \alpha y^2 = -g(u) \tag{10}$$

with $z = y(v + \frac{1}{2}u + \frac{1}{2}a_2)$ and $\alpha = \frac{3}{4}u^2 + \frac{1}{2}a_2u + a_4 - \frac{1}{4}a_2^2$. This is a non degenerate conic as soon as α and $g(u)$ are both non zero (which happens for all $u \in \mathbb{F}_q$ except at most 5), and then admits a rational parametrization, yielding a rational curve $\mathbb{A}^1 \to S$. Composing with ψ, we get the required rational curve on V, and hence an encoding, provided that $q > 5$.

To make the encoding function entirely explicit, however, it is necessary to find an explicit $\mathbb{F}_q$-point on the conic (10). It is easy to find such a point using a randomized algorithm for any given set of parameters, and Shallue and van de Woestijne even propose an algorithm to do so in *deterministic* polynomial time, but it seems difficult to give a convenient expression for that point, and hence the encoding, in the general case.

BN curves (8), however, correspond to the special case when $a_2 = a_4 = 0$ and $a_6 = b$, where Shallue and van de Woestijne's formulas simplify significantly. Since we are given the point G of abscissa 1 on the curve, it seems convenient to pick $u = 1$. Then, the conic (10) becomes:

$$z^2 + \frac{3}{4}y^2 = -1 - b = -y_{\mathsf{G}}^2. \tag{11}$$

Moreover, since the base is field $\mathbb{F}_q = \mathbb{F}_p$ has $p \equiv 1 \pmod 3$ elements, -3 is a quadratic residue, as is easily seen either by quadratic reciprocity of by expressing $\sqrt{-3}$ in terms of a cube root of unity. At any rate, we obtain an explicit expression for a point on (11), namely:

$$(z; y) = \left(0\,;\ \frac{2y_{\mathsf{G}}}{\sqrt{-3}}\right).$$

This makes it possible to write a completely explicit rational parametrization of the conic (11), and if $(z; y)$ is a point on that conic, we know from the expression of ψ that at least one of $x_0 = z/y - 1/2$, $x_1 = -1 - x_0$ or $x_2 = 1 + y^2$ is the abscissa of a point on the BN curve (8). This gives a convenient encoding function, especially in the case $p \equiv 3 \pmod 4$ (recommended by Barreto and Naehrig) in which square roots are easy to compute.

5.4 Limitations and perspectives

There are several ways in which the encoding described above could be improved further. For one, while it is relatively easy to write down a constant-time implementation of that encoding (to protect against timing attacks), it is less obvious how to implement it without branching (to protect against glitch attacks). Moreover, the natural implementation computes two Legendre symbols and one exponentiation; if the implementer wants to avoid Legendre symbols (for the reasons mentioned at the end of §3), implementing it using fewer than two full exponentiations is an open problem, as is the question of minimizing divisions.

Another important problem is indifferentiable hashing on BN curves, in the sense of [10]. In principle, the techniques introduced by Farashahi et al. in [14] should apply to this case, but there are a number of possible technical difficulties, including the fact that we are making arbitrary choices on which abscissa x_i to use, and the relatively high degree of the function field extensions involved. The same techniques, if they apply, should also provide a precise evaluation of the number of points in the image of the encoding, and of its collision probability.

Finally, even if it does simplify to some extent in the case of BN curves, the approach of Shallue and van de Woestijne seems quite complex in comparison with other proposed constructions of elliptic curve encodings. The exceedingly simple form of BN curves strongly suggests that a simpler approach should exist for that case as well, but finding such remains an intriguing open question.

References

[1] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2004.

[2] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[3] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

[4] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[5] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[6] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[7] Colin Boyd, Paul Montague, and Khanh Quoc Nguyen. Elliptic curve based password authenticated key exchange protocols. In Vijay Varadharajan and Yi Mu, editors, *ACISP*, volume 2119 of *Lecture Notes in Computer Science*, pages 487–501. Springer, 2001.

[8] Xavier Boyen. Multipurpose identity-based signcryption (a Swiss army knife for identity-based cryptography). In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2003.

[9] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2000.

[10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into

ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 237–254. Springer, 2010.

[11] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2003.

[12] Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 511–526. Springer, 2005.

[13] Jean-Marc Couveignes and Jean-Gabriel Kammerer. The geometry of flex tangents to a cubic curve and its parameterizations. Cryptology ePrint Archive, Report 2011/033, 2011. http://eprint.iacr.org/.

[14] Reza R. Farashahi, Pierre-Alain Fouque, Igor E. Shparlinski, Mehdi Tibouchi, and J. Felipe Voloch. Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. *Math. Comput.*, 2012. To appear.

[15] Reza Rezaeian Farashahi. Hashing into hessian curves. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT*, volume 6737 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2011.

[16] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[17] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.

[18] Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 303–316. Springer, 2009.

[19] David P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, 26:5–26, October 1996.

[20] David P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *WETICE*, pages 248–255. IEEE Computer Society, 1997.

[21] Jean-Gabriel Kammerer, Reynald Lercier, and Guénaël Renault. Encoding points on hyperelliptic curves over finite fields in deterministic polynomial time. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 2010.

[22] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2004.

[23] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[24] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

[25] Hisayoshi Sato and Keisuke Hakuta. An efficient method of generating rational points on elliptic curves. *J. Math-for-Industry*, 1(A):33–44, 2009.

[26] Andrejz Schinzel and Mariusz Skałba. On equations $y^2 = x^n + k$ in a finite field. *Bull. Pol. Acad. Sci. Math.*, 52(3):223–226, 2004.

[27] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44(170):483–494, 1985.

[28] Andrew Shallue and Christiaan van de Woestijne. Construction of rational points on elliptic curves over finite fields. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 510–524. Springer, 2006.

[29] Mariusz Skałba. Points on elliptic curves over finite fields. *Acta Arith.*, 117:293–301, 2005.

[30] Maciej Ulas. Rational points on certain hyperelliptic curves over finite fields. *Bull. Pol. Acad. Sci. Math.*, 55(2):97–104, 2007.

[31] Fangguo Zhang and Kwangjo Kim. ID-based blind signature and ring signature from pairings. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer, 2002.