Implementation of the Boneh-Franklin IBE Scheme

Florence Lam*

Project advisor: Gabriel Dorfsman-Hopkins†

Abstract

In this paper and accompanying software, we give a fully functional implementation of the Boneh-Franklin Identity-Based Encryption (IBE) scheme using the Weil pairing, which runs efficiently even with primes of cryptographic size. We describe the conceptual framework of the IBE, give background on the Weil pairing. Further, we discuss the challenges in the process of creating a functional implementation, and how we overcame them. The reader is encouraged to experiment with the accompanying software, which is written in SageMath.

1 Introduction

Elliptic curve cryptography was invented independently by Miller [6] in 1985 and Koblitz [4] in 1987. Koblitz developed elliptic curve cryptosystems because they rely on the hardness of the elliptic curve discrete logarithm problem (ECDLP), which is likely harder than the traditional discrete logarithm problem (DLP) [4]. In addition to increased security, elliptic curve systems are more efficient than Diffie-Hellman schemes over finite fields [6].

Typical cryptographic schemes require Alice and Bob to exchange either public or private keys. However, Identity-Based Encryption (IBE) schemes, proposed by Shamir [7] in 1984, do not require key exchange. Alice can use any combination of information that pertains to Bob's identity (such as his birthdate, name, or email address) to encrypt her message. Once Bob receives Alice's ciphertext, he decrypts the ciphertext with the private key that the Public Key Generator (PKG) computes using his information. It was an open problem until Boneh and Franklin [1] proposed a pairing-based scheme and Cocks proposed a scheme based on quadratic reciprocity [2]. In this article, we implement the Boneh-Franklin scheme using Sage and discuss the challenges we faced while implementing this scheme. A complete implementation written in Sage is available in GitHub [5].

2 Background

In this section, we first discuss the Weil pairing, a pairing on m-torsion points in an elliptic curve E. Next, we introduce an algorithm to efficiently compute the Weil pairing. And finally, we explain the algorithms involved in the Boneh-Franklin IBE scheme, and why we must use a modified Weil pairing for this scheme.

2.1 Bilinear Pairings, Rational Functions, and Divisors

Before introducing the Weil pairing, we need to understand what bilinear pairings, rational functions, and divisors are.

*University of California, Berkeley, Department of Mathematics, florencelam@berkeley.edu

†Saint Lawrence University, Department of Mathematics, Statistics and Computer Science, gdhopkins@stlawu.edu

Definition 2.1. Let K be a field and V be a vector space over K. A *bilinear pairing* is a function $\beta\colon V\times V\to K$ that satisfies the following:

1. $\beta(c_1\mathbf{u}+c_2\mathbf{v},\mathbf{w})=c_1\beta(\mathbf{u},\mathbf{w})+c_2\beta(\mathbf{v},\mathbf{w})$
2. $\beta(\mathbf{u},c_1\mathbf{v}+c_2\mathbf{w})=c_1\beta(\mathbf{u},\mathbf{v})+c_2\beta(\mathbf{u},\mathbf{w}),$

where $c_1,c_2\in K$, and $\mathbf{u},\mathbf{v},\mathbf{w}\in V$.

Definition 2.2. A *rational function* is a ratio of polynomials

$$f(x)=\frac{a_0+a_1x+\cdots+a_nx^n}{b_0+b_1x+\cdots+b_mx^m} \tag{1}$$

over a field K.

From the fundamental theorem of algebra, any nontrivial polynomial can be completely factored over an algebraic closure, which means that we can write any rational function as

$$f(x)=\frac{a(x-\alpha_1)^{e_1}(x-\alpha_2)^{e_2}\cdots(x-\alpha_s)^{e_s}}{b(x-\beta_1)^{d_1}(x-\beta_2)^{d_2}\cdots(x-\beta_r)^{d_r}}, \tag{2}$$

where $\sum t_i=n$ and $\sum d_j=m$, and we cancel and combine like terms so that every a_i and b_i is unique. We call a_i, the roots of the polynomial on the numerator, the *roots* of the rational function $f(x)$, and we call b_j, the roots of the polynomial on the denominator, the *poles* of $f(x)$. The exponents e_k and d_ℓ are the *multiplicities*. Zeros, poles, and multiplicities are all properties of $f(x)$, and we can tally up these values using a *divisor*.

Definition 2.3. Let $f(x)$ be the rational function in 2. The *divisor* of $f(x)$ is the formal sum

$$div(f)=e_1[\alpha_1]+e_2[\alpha_2]+\cdots+e_s[\alpha_s]-d_1[\beta_1]-d_2[\beta_2]\cdots-d_r[\beta_r]. \tag{3}$$

Note that this formal sum is simply a convenient way to say that $f(x)$ has a zero of multiplicity e_1 at α_1, a zero of multiplicity e_2 at α_2, etc.

The definitions above apply to a rational function with only one variable. However, the Weil pairing works with rational functions on elliptic curves that have two variables x,y. For example, suppose we have a nontrivial rational function $f(x,y)$ on an elliptic curve $E: y^2=x^3+Ax+B$. A point $P\in E$ has the form $P=(x,y)$, so we can write $f(x,y)$ as $f(P)$. Like any rational function, $f(P)$ may have zeros or poles at certain points in E. We can again use the divisor to note whether or not a point in E is a zero or pole counting multiplicity.

Definition 2.4. Let f be a rational function on an elliptic curve E. For each point $P\in E$, let n_P be the multiplicity of the vanishing of f at P, where we write $-n_P$ to denote the multiplicities of poles. The *divisor* of f is

$$div(f)=\sum_{P\in E}n_P[P].$$

Since f has finitely many zeros and poles, only finitely many of the coefficients n_P are nonzero. While we expanded our definition of the divisor to make sense for rational functions on E, we can further generalize the definition which does not require a function.

Definition 2.5. A *divisor* D on an elliptic curve E is a formal sum of the form

$$D=\sum_{P\in E}n_P[P].$$

Definition 2.6. The *degree* of a divisor is the sum of the coefficients n_P

$$deg(D) = \sum_{P \in E} n_P,$$

and the *sum* of a divisor simply adds up multiples of points using the group law on E

$$sum(D) = \sum_{P \in E} n_P P.$$

Since this definition does not even require a function, we might be curious whether we can determine whether a divisor is the divisor of some function, and in our case, a rational function. Luckily, there is a theorem that tells us a way to determine whether a divisor is the divisor of a rational function.

Theorem 2.1. Let E be an elliptic curve, and let D be a divisor on E.

1. Let f and g be nontrivial rational functions on E. Then $div(f) = div(g)$ (i.e. f and g share the same zeros, poles, and associated multiplicities) if and only if there is a constant c such that $f = cg$.
2. D is the divisor of a rational function on E if and only if $deg(D) = 0$ and $sum(D) = \mathcal{O}$, where $\mathcal{O}$ is the point of infinity.

Proof. Please refer to Propositions III.3.1 and III.3.4 in pages 59 and 61 of [8]. □

Part 1 of Theorem 2.1 tells us that a rational function with a given divisor is unique up to some constant. Part 2 is an interesting result because we can determine the existence of a rational function by adding points and adding multiplicities. However, it does not find us an exact rational function corresponding to a divisor.

2.2 A Special Divisor and Miller's Algorithm

For the Weil pairing, we are primarily concerned with a special type of divisor that has the form $D = m[P] - m[\mathcal{O}]$, where $P \in E[m]$ (note: $E[m]$ is the set of m-torsion points on E) is a point with order m. From Theorem 2.1, we know that D is the divisor of some rational function since $deg(D) = m - m = 0$ and $sum(D) = mP - \mathcal{O} = \mathcal{O}$. The problem that arises is whether we can concretely find a rational function f_P such that $div(f_P) = m[P] - m[\mathcal{O}]$. It turns out Miller's Algorithm is a fast algorithm that can find rational functions with divisors of the form $m[P] - [mP] - (m-1)[\mathcal{O}]$, where $P \in E$ is not necessarily an m-torsion point. In particular, if $P \in E[m]$, Miller's algorithm will return a rational function f_P with $div(f_P) = m[P] - m[\mathcal{O}]$, since $m[P] - [mP] - (m-1)[\mathcal{O}] = m[P] - m[\mathcal{O}]$.

Theorem 2.2. Let E be an elliptic curve and let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be nonzero points in E [3].

1. Suppose λ is the slope of the line that passes through P and Q (if $Q = P$, let λ be the slope tangent to E at P, and if the line is vertical, let $\lambda = \infty$). Let the function $g_{P,Q}$ be the following:

$$g_{P,Q} = \begin{cases} \dfrac{y - y_P - \lambda(x - x_P)}{x + x_P + x_Q - \lambda^2} & \text{if } \lambda \neq \infty, \\ x - x_P & \text{if } \lambda = \infty. \end{cases}$$

   Then

$$div(g_{P,Q}) = [P] + [Q] - [P + Q] - [\mathcal{O}].$$

2. Let $m \geq 1$ have the following binary expansion:

$$m = m_0 + m_1 \cdot 2^1 + m_2 \cdot 2^2 + \cdots + m_{n-1} \cdot 2^{n-1},$$

   where $m_i \in 0, 1$ and $m_{n-1} = 1$. *Miller's Algorithm* returns a rational function f_P with divisor

$$div(f_P) = m[P] - [mP] - (m-1)[\mathcal{O}].$$

Proof. Please refer to Section XI.8 of [8] for a complete proof of correctness. □

Algorithm 1: Miller's Algorithm (from [3] page 344)

Input: P, E, n
Output: the value f

Set $T = P$ and $f = 1$
for $i \leftarrow n - 2$ **to** 0 **do**
  Set $f = f^2 \cdot g_{T,T}$
  Set $T = 2T$
  if $m_i = 1$ **then**
    Set $f = f \cdot g_{T,P}$
    Set $T = T + P$
end

2.3 The Weil Pairing and its Special Properties

Definition 2.7. Let $P, Q \in E[m]$, and let f_P and f_Q be rational functions on E with divisors $div(f_P) = m[P] - m[\mathcal{O}]$ and $div(f_Q) = m[Q] - m[\mathcal{O}]$, respectively. The *Weil pairing* e_m of P and Q is

$$e_m(P, Q) = \frac{f_P(Q+S)}{f_P(S)} / \frac{f_Q(P-S)}{f_Q(-S)}, \tag{4}$$

where $S \in E$ is a point such that $S \notin \{\mathcal{O}, P, -Q, P - Q\}$.

To compute the Weil pairing, we simply use Miller's Algorithm to find the rational functions f_P and f_Q, then evaluate the functions. The Weil pairing does not appear to be well-defined because we are choosing the functions f_P and f_Q and the point S. However, Theorem 2.3 shows that the Weil pairing is well-defined and does not depend on the choice of S.

Theorem 2.3. Let e_m be a Weil pairing. Then we have the following:

1. The value of the Weil pairing does not depend on the choice of f_P, f_Q, and S.
2. The value of the Weil pairing is an mth root of unity, that is, it satisfies $e_m(P, Q)^m = 1$ for every $P, Q \in E[m]$.
3. The Weil pairing is a billinear pairing:

$$e_m(P_1 + P_2, Q) = e_m(P_1, Q)e_m(P_2, Q) \text{ for all } P_1, P_2, Q \in E[m] \tag{5}$$

   and

$$e_m(P, Q_1 + Q_2) = e_m(P, Q_1)e_m(P, Q_2) \text{ for all } P, Q_1, Q_2 \in E[m]. \tag{6}$$

4. The Weil pairing is alternating: $e_m(P, P) = 1$ for all $P \in E[m]$.
5. The Weil pairing is nondegenerate: if $e_m(P, Q) = 1$ for all $Q \in E[m]$, then $P = \mathcal{O}$.

Proof. For a complete proof of the five parts, see [8] Section III.8. □

One thing to note is that the Weil pairing is a bilinear pairing, but instead of addition on the right hand side of (4), we have multiplication. In particular, if we have an integer a, then

$$\begin{aligned} e_m(aP, Q) &= \underbrace{e_m(P, Q) \cdot \ldots \cdot e_m(P, Q)}_{a \text{ times}} \\ &= e_m(P, Q)^a. \end{aligned}$$

So, instead of multiplying integers with the bilinear pairing, we exponentiate. Bilinearity is important in the IBE scheme because it allows Bob to decrypt Alice's ciphertext.

2.4 A General IBE Scheme

Here we describe a general IBE Scheme called BasicIdent. We can categorize BasicIdent into four parts: Setup, Extract, Encrypt, Decrypt. Let $k \geq 2 \in \mathbb{Z}$ be a security parameter input for the Setup algorithm, and let $\mathcal{G}$ be a BDH parameter generator as described in [1].

Setup: Choose a security parameter $k \geq 2 \in \mathbb{Z}$. The Setup algorithm does the following:

1. Choose some k and input into $\mathcal{G}$ to generate primes p, q, two different groups $\mathbb{G}_1, \mathbb{G}_2$ both with order q, and a bilinear pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Also, choose a generator $P \in \mathbb{G}_1$.
2. Choose a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.
3. Choose two hash functions H_1 and H_2, let $\mathcal{M}$ be the message space, and let $\mathcal{C}$ be the ciphertext space.
4. The output is the system parameters $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$. Note that s is the master key, which only the PKG knows.

Extract: Given a string ID that pertains to Bob's identity, this algorithm does the following:

1. Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1$
2. Compute the private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. Bob will use d_{ID} to decrypt Alice's ciphertext.

Encrypt: Using Q_{ID} to encrypt a message $M \in \mathcal{M}$, this algorithm does the following:

1. Choose a random $r \in \mathbb{Z}_q^*$
2. Compute $g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2$
3. Compute the ciphertext $C = \langle rP, M \oplus H_2(g_{\mathsf{ID}}^r) \rangle$.

Decrypt: Let $C = \langle U, V \rangle \in \mathcal{C}$ be the ciphertext generated by Encrypt. We can use the private key d_{ID} to decrypt C by computing $V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M$.

Proof. Here, we prove that the Decrypt algorithm is correct. Because of bilinearity and the fact that $P_{pub} = sP$, the pairing gives us $\hat{e}(d_{\mathsf{ID}}, U) = \hat{e}(sQ_{\mathsf{ID}}, rP) = \hat{e}(Q_{\mathsf{ID}}, P)^{rs} = \hat{e}(Q_{\mathsf{ID}}, P_{pub})^r = g_{\mathsf{ID}}^r$. Decrypt computes $V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M \oplus H_2(g_{\mathsf{ID}}^r) \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M \oplus H_2(g_{\mathsf{ID}}^r) \oplus H_2(g_{\mathsf{ID}}^r) = M$. □

2.5 The Boneh-Franklin Scheme

In the previous section, we mentioned the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. But what are $\mathbb{G}_1$ and $\mathbb{G}_2$, and what pairing will we use? It will turn out that $\mathbb{G}_1$ will be q-torsion points on a specially chosen elliptic curve for a specific prime value of q, and the pairing will be a modified form of the Weil pairing. As we saw in Theorem 2.3.2, this takes values in qth roots of unity, so this will be $\mathbb{G}_2$. How do we map messages to q-torsion points of an elliptic curve? How do we modify the Weil pairing, and why do we have to do this? We will discuss the concrete IBE scheme in this section, and answer the questions we have posed above.

The first part of BasicIdent is the Setup algorithm. This algorithm generates the primes, groups, and public keys that form the basis of the three later algorithms. In the concrete IBE scheme, the BDH parameter $\mathcal{G}_1$ is the Setup algorithm, while the composition of a hash function and the MapToPoint function forms the Extract algorithm. Both Encrypt and Decrypt in the concrete IBE scheme are same as the ones in BasicIdent. Below is a summary of the algorithms unique to the Boneh-Franklin scheme:

BDH Parameter Generator $\mathcal{G}_1$: We input a security parameter $2 < k \in \mathbb{Z}^+$ which determines the size of the keys. The generator $\mathcal{G}_1$ picks a k-bit prime q and the smallest prime p satisfying (1) $p = 2 \mod 3$, (2) $q \mid p + 1$, and (3) $q^2 \nmid p + 1$. With the two primes, we can generate the two groups. The group $\mathbb{G}_1 = E[q](\mathbb{F}_p) = \langle P \rangle$ is a subgroup of order q generated by the point P on the elliptic curve $E(\mathbb{F}_p) : y^2 = x^3 + 1$. The group $\mathbb{G}_2$ is a subgroup of $\mathbb{F}_{p^2}^*$ with order q, and the modified Weil pairing is the map $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ defined below.

Definition 2.8. Let $1 \neq \zeta \in \mathbb{F}_p$ be a solution of $x^3 - 1 = 0$ in $\mathbb{F}_{p^2}$. The *modified Weil pairing* $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a map that does the following:

$$\hat{e}(P, Q) = e(P, \phi(Q)),$$

where $\phi(Q) = \phi(x_Q, y_Q) = (\zeta x_Q, y_Q)$.

Hash Function H_1 and MapToPoint: Suppose we have the primes p satisfying the three properties above and q. Let $p = \ell q + 1$ for some $\ell \in \mathbb{Z}$, and choose a string ID which pertains to Bob's identity. Let $y_0 = H_1(\mathsf{ID}) \in \mathbb{F}_p$ be the output of H_1 and the input of MapToPoint. The MapToPoint algorithm does the following:

1. Find $x_0 = (y_0^2 - 1)^{1/3} \in \mathbb{F}_p$
2. Let $Q = (x_0, y_0) \in E(\mathbb{F}_p)$
3. Return $Q_{\mathsf{ID}} = \ell Q \in \mathbb{G}_1$.

The two algorithms seem fairly simple; however, there are some questions that arise:

(i) Can we always find a subgroup $\mathbb{G}_1$ of order q?

(ii) Why is it necessary to use the modified Weil pairing?

(iii) Is there always a solution to $x_0 = (y_0^2 - 1)^{1/3} \mod p$ for any $y_0 \in \mathbb{F}_p$? If so, is it unique?

We answer these three questions now, beginning with the first:

Can we always find a subgroup $\mathbb{G}_1$ of order q? From Exercise 10.19 of [8], the order of $E : y^2 = x^3 + 1$ over $\mathbb{F}_p$ is $p + 1$. And since $q \mid p + 1$, Cauchy's Theorem tells us that $E(\mathbb{F}_p)$ must have a subgroup of order q. Thus, the generator $\mathcal{G}_1$ will always find such a subgroup.

Why is it necessary to use the modified Weil pairing? If we use the usual Weil pairing in the BDH Generator, then we have $e\colon E[q] \times E[q] \to \mathbb{G}_2$. Say we have two points $R, S \in E[q]$. Since $E[q] = \langle P \rangle$, there are two integers r, s such that $R = rP$ and $S = sP$. Because the pairing is bilinear and alternating, we get $e(R, S) = e(rP, sP) = e(P, P)^{rs} = 1$. This means that the regular pairing is degenerate on $E[q]$. This field extension is very important because it ensures that $\phi(Q) \notin E(\mathbb{F}_p)$.

Is there always a solution to $x_0 = (y_0^2 - 1)^{1/3} \mod p$ in MapToPoint? If so, is x_0 unique? It turns out that the answer to both questions is yes, according to Exercise 3.41 of [3].

3 Implementing the Boneh-Franklin Scheme

In this section, we describe how to implement the Boneh-Franklin IBE Scheme and the modified Weil pairing using Sage. We discuss some challenges we had while implementing. Finally, we ask and answer some questions related to these algorithms.

A general identity-based encryption scheme has four algorithms:

1. Setup
2. Extract
3. Encrypt
4. Decrypt

As we have mentioned in the previous section, the BDH parameter generator $\mathcal{G}_1$ is the Setup algorithm, while the Extract function is MapToPoint composed with H_1, a hash function.

3.1 The BDH parameter generator $\mathcal{G}_1$

The goal of the BDH parameter generator $\mathcal{G}_1$ (algorithm shown below) is to find primes p, q, generate an elliptic curve over $\mathbb{F}_p$, and find a point P with order q. These outputs are important because they allow for encryption and decryption.

Algorithm 2: BDH parameter generator $\mathcal{G}_1$

Input: $2 < k \in \mathbb{Z}^+$
Output: params, s

1 Find a random k-bit prime q
2 Set $p = q$ /* Find prime p using Lines 2-11 */
3 Set $\ell = 1$
4 Set $\ell q = q$
5 **while** *true* **do**
6 | **repeat**
7 | | Set $\ell = \ell + 1$
8 | | Set $\ell q = \ell q + q$
9 | | Set $p = \ell q - 1$
10 | **until** $p \equiv 2 \mod 3$, $q \mid p+1$, $q^2 \nmid p+1$, and p is prime.
11 **end**
12 Set $E : y^2 = x^3 + 1 \mod p$
13 Set $P = \mathcal{O}$ /* Find point P with order q using Lines 12-20 */
14 **while** $P = \mathcal{O}$ **do**
15 | Set Q to be a random point in E
16 | **while** $Q = \mathcal{O}$ **do**
17 | | Set Q to be a random point in E
18 | **end**
19 | Set $h = \frac{p+1}{q}$
20 | Set $P = hQ$
21 **end**
22 Set s to be a random number in $\mathbb{Z}_q^*$ /* s is the master key */
23 Set $P_{pub} = sP$
24 Set params $= \langle p, q, \ell, E, P, P_{pub} \rangle$

For the first step, we found the random k-bit prime q using a built-in pseudo-random integer generator from Sage; however, we struggled finding a prime p satisfying (1) $p \equiv 2 \mod 3$, (2) $q \mid p+1$, and (3) $q^2 \nmid p+1$. We first tried by using the `random_prime` function to find random primes between q and q^2, and checked if the prime satisfied all three properties. This method successfully found a prime p, but in practice, it was slow. For our second try, instead of testing random primes, we noticed that property (2) would be useful for finding p. Since $q \mid p+1$, there is some $\ell \in \mathbb{Z}$ such that $p + 1 = \ell q$. We started with $\tilde{\ell} = 2$ and calculated $\tilde{\ell} q$, subtracted it by 1 to get $\tilde{p} = \tilde{\ell} q - 1$, a candidate for p, and then tested whether $\tilde{p}$ satisfied (1) to (3). If $\tilde{p}$ did not satisfy the three properties, we increased $\tilde{\ell}$ by 1 until we found a $\tilde{p}$ that satisfied the three properties, and set $p = \tilde{p}$. From our experience, the second method found p faster than the first method, but why exactly is the second method faster? We will show why the second method is faster by calculating expected numbers, but before we explain this, here is a useful theorem:

Theorem 3.1. If p is the probability of success on each trial, then the expected number of trials until the first success is $1/p$.

The first method finds a random prime $\tilde{p}$ between q and q^2, and checks whether $\tilde{p}$ satisfies (1) and (2). To find the expected number, let us consider two events A and B:

$$A = \{n \colon n \equiv 2 \mod 3\}$$
$$B = \{n \colon q \mid n+1\}.$$

Any prime $\tilde{p} \neq 3$ must satisfy $\tilde{p} \equiv 1, 2 \mod 3$, so $P(A) = 1/2$. Additionally, $P(B) = 1/q$. Combining these two probabilities gives us the probability that $\tilde{p}$ satisfies (1) and (2), which is

$$P(A \cap B) = P(A) \times P(B) \tag{7}$$
$$= \frac{1}{2q}. \tag{8}$$

Thus by Theorem 3.1, the expected number of trials until finding p with the first method is $E_1 = \frac{1}{P(A \cap B)} = 2q$.

On the other hand, the second method finds a multiple $q\tilde{\ell}$ of q between q and q^2, and checks whether $q\tilde{\ell} - 1$ is prime and whether it satisfies (1). Consider two events A and C:

$$A = \{n \colon n \equiv 2 \mod 3\}$$
$$C = \{q < n < q^2 \colon n \text{ is prime}\}.$$

To estimate $P(C)$, we use the prime number theorem to estimate the number of primes between $q\tilde{\ell} - 1$ and $q(\tilde{\ell}+1) - 1$:

$$\pi(q(\tilde{\ell}+1) - 1) - \pi(q\tilde{\ell} - 1) \approx \pi(q(\tilde{\ell}+1)) - \pi(q\tilde{\ell}) \tag{9}$$
$$\approx \frac{q(\tilde{\ell}+1)}{\log q(\tilde{\ell}+1)} - \frac{q\tilde{\ell}}{\log q\tilde{\ell}}. \tag{10}$$

Since $\log q(\tilde{\ell}+1) \approx \log q(\tilde{\ell})$, we have

$$\frac{q(\tilde{\ell}+1)}{\log q(\tilde{\ell}+1)} - \frac{q\tilde{\ell}}{\log q\tilde{\ell}} \approx \frac{q(\tilde{\ell}+1) - q\tilde{\ell}}{\log q\tilde{\ell}} \tag{11}$$
$$= \frac{q}{\log q\tilde{\ell}} \tag{12}$$
$$\gg 1 \tag{13}$$

whenever $\tilde{\ell} \ll \frac{e^q}{q}$. This tells us that if $\tilde{\ell} \ll \frac{e^q}{q}$, the number of primes between consecutive multiples of q is much greater than 1. Thus, the approximate probability that a random number between q and q^2 is prime is $P(C) \approx \frac{1}{\log q\tilde{l}}$. In general, we can assume $\tilde{l} \ll q$ since primes are quite common, so $P(C) \approx \frac{1}{\log q}$. The probability that we find a prime p using the second method is

$$P(A \cap C) = P(C) \times P(A \mid C) \tag{14}$$
$$\approx \frac{1}{\log q} \times \frac{1}{2} \tag{15}$$
$$= \frac{1}{2\log q}. \tag{16}$$

Theorem 3.1 tells us that the expected number of trials until finding p with the second method is $E_2 = \frac{1}{P(A \cap C)} = 2\log q$. If we compare E_1 and E_2, it is clear that $E_2 \ll E_1$, which means that in general, the second method is much faster than the first.

The last section of Algorithm 2 finds a point $P \in E[q](\mathbb{F}_p)$. It starts by randomly selecting a point $Q \in E \setminus \{\mathcal{O}\}$, and then setting $P = hQ$, where $h = \frac{p+1}{q}$. This generates a point P with order q as long as $P \neq \mathcal{O}$ by Lemma 3.1.

Lemma 3.1. Let $E(\mathbb{F}_p)$ be an elliptic curve over $\mathbb{F}_p$, and let $\mathcal{O} \neq Q \in E(\mathbb{F}_p)$. If $P \neq \mathcal{O}$ and $P = hQ$, where $h = \frac{p+1}{q}$, then P is a point of order q.

Proof. Let E be the elliptic curve $y^2 = x^3 + 1 \mod p$ and let p be the prime satisfying conditions in line 10 of Algorithm 2. By Exercise 10.19 of [8], the number of points in E is $p+1$. Suppose we have two points $P, Q \in E$ as above. Then $qP = q(hQ) = q\left(\frac{p+1}{q}P\right) = (p+1)P = \mathcal{O}$, since $\#E(\mathbb{F}_p) = p+1$. By Lagrange's Theorem, the order of P divides q, which means we have two possibilities:

$$|P| = \begin{cases} 1 & \text{if } Q = \mathcal{O}, \\ q & \text{otherwise.} \end{cases}$$

However, the point $Q \neq \mathcal{O}$, so $|P| = q$.

□

3.2 MapToPoint: An algorithm mapping strings to points

The input of the MapToPoint algorithm is the output of the hash algorithm H_1. Using both H_1 and MapToPoint, we can map the string ID to a point on the elliptic curve generated by Algorithm 2. Here is an outline of the algorithm:

Algorithm 3: MapToPoint

Input: params and $y_0 \in \mathbb{F}_p$, where y_0 is an output of the hash algorithm H_1
Output: $d_{\mathsf{ID}}, Q_{\mathsf{ID}} \in \mathbb{G}_1^*$

1 Set $p, q, \ell, E, P, P_{pub} = \mathsf{params}$
2 Set $x_0 = (y_0^2 - 1)^{\frac{2p-1}{3}} \mod p$
3 Set $Q = (x_0, y_0) \in E(\mathbb{F}_p)$
4 Set $Q_{\mathsf{ID}} = \ell Q$
5 Set $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$

Recall from Section 2.5 that MapToPoint computes $x_0 = (y_0^2 - 1)^{1/3}$. In the algorithm above, we instead set $x_0 = (y_0^2 - 1)^{\frac{2p-1}{3}}$. This works because of Euler's theorem:

$$\begin{aligned}
(y_0^2 - 1)^{\frac{2p-1}{3}} &= (y_0^2 - 1)^{\frac{2(p-1)+1}{3}} \\
&= (y_0^2 - 1)^{\frac{2(p-1)}{3}} (y_0^2 - 1)^{\frac{1}{3}} \\
&= \left((y_0^2 - 1)^{p-1}\right)^{\frac{2}{3}} (y_0^2 - 1)^{\frac{1}{3}} \\
&= \left((y_0^2 - 1)^{\phi(p)}\right)^{\frac{2}{3}} (y_0^2 - 1)^{\frac{1}{3}} \\
&= (y_0^2 - 1)^{\frac{1}{3}} \\
&= x_0.
\end{aligned}$$

We compute $(y_0^2 - 1)^{\frac{2p-1}{3}}$ because $p = 2 \mod 3$. This means that $3 \mid (2p-1)$, and we can compute x_0 efficiently using the fast powering algorithm in Exercise 1.25 of [3].

For the Encrypt and Decrypt algorithms, we had to create a function that computes the modified Weil pairing. The key part to this function is $\zeta \neq 1 \in \mathbb{F}_p$, which is a solution of $x^3 - 1 = 0$ in $\mathbb{F}_{p^2}$. Note that the polynomial $x^3 - 1 = (x-1)(x^2 + x + 1)$, and since 1 is a root of $x - 1$, ζ must be a root of $x^2 + x + 1$. More precisely, the field $\mathbb{F}_{p^2} \cong \mathbb{F}_p[x]/\langle x^2 + x + 1\rangle$. Other than defining ζ in the modified Weil pairing, both algorithms are relatively simple and we followed the steps in BasicIdent.

3.3 Acknowledgement

I would like to thank my mentor, Professor Gabriel Dorfsman-Hopkins, for his support throughout this project. He is the one who introduced me to cryptography, encouraged me when I doubted myself, and reached out when I felt like I did not belong in math. Without him, I would not have been able to explore my interests.

I would also like to thank my former classmates, Jennifer Buettner and Toren Warady, for helping me learn how to code. They guided me when I struggled and gave me new perspectives on solving problems. It is because of them that I learned the joy of collaboration.

References

[1] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[2] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and coding*, volume 2260 of *Lecture Notes in Comput. Sci.*, pages 360–363. Springer, Berlin, 2001.

[3] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An introduction to mathematical cryptography.* Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.

[4] Neal Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.

[5] Florence Lam. Boneh-Franklin IBE scheme. `https://github.com/gdorfsmanhopkins/BonehFranklinIBE`, 2022.

[6] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology—CRYPTO '85 (Santa Barbara, Calif., 1985)*, volume 218 of *Lecture Notes in Comput. Sci.*, pages 417–426. Springer, Berlin, 1986.

[7] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology (Santa Barbara, Calif., 1984)*, volume 196 of *Lecture Notes in Comput. Sci.*, pages 47–53. Springer, Berlin, 1985.

[8] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1986.