The Practical Advantage of RSA over ECC and Pairings

Zhengjun Cao and Lihua Liu

Abstract. The coexistence of RSA and elliptic curve cryptosystem (ECC) had continued over forty years. It is well-known that ECC has the advantage of shorter key than RSA, which often leads a newcomer to assume that ECC runs faster. In this report, we generate the Mathematica codes for RSA-2048 and ECC-256, which visually show that RSA-2048 runs three times faster than ECC-256. It is also estimated that RSA-2048 runs 48,000 times faster than Weil pairing with 2 embedding degree and a fixed point.

Keywords: RSA, ECC, Weil pairing, embedding degree

1 Introduction

The public key cryptosystem RSA was published by Rivest, Shamir and Adleman [8] in 1978. Koblitz [5] and Miller [6] in 1985 independently proposed using the group of points on an elliptic curve over a finite field to devise discrete logarithm cryptographic schemes. Pairing based cryptography, introduced by Boneh and Franklin [2], had also been intensively studied over twenty years. Although Shor's algorithm [9] for factorization was regarded as a big threat to RSA, the current toy quantum machine including IBM 1,000-qubit quantum chip [3] still cannot be used to test Shor's algorithm.

So far, the fastest algorithm known for factorization or for general discrete logarithm problem is the Number Field Sieve (NFS) which has a subexponential expected running time of

$$O(e^{(1.923+o(1))(\log\, n)^{1/3}(\log\, \log\, n)^{1-1/3}})$$

The fastest algorithm known for elliptic curve discrete logarithm problem (ECDLP) is Pollard's rho algorithm which has an expected running time of $\frac{\sqrt{\pi\, n}}{2}$. We refer to the below Table 1 for RSA, Discrete Logarithm (DL) and Elliptic Curve (EC) key sizes for equivalent security levels [4].

Table 1: Different key sizes for equivalent security levels

security level (bits)	80	112	128	192	256
RSA modulus n (modulus)	1024	2048	3072	8192	15360
DL parameter q (order)	160	224	256	384	512
EC parameter n (order)	160	224	256	384	512

ECC-256 can provide the same security level as RSA-2048. The surprising advantage of ECC has attracted much attention. But we have noticed that ECC has not yet replaced RSA. How long will the coexistence of RSA and ECC last?

Z. Cao is with Department of Mathematics, Shanghai University, Shanghai, China.
L. Liu is with Department of Mathematics, Shanghai Maritime University, Shanghai, China. Email: liulh@shmtu.edu.cn

The well-known advantage of ECC often leads a newcomer to mistakenly assume that ECC runs faster. In this report, we generate the Mathematica codes to test RSA-2048 and ECC-256. The results visually show that RSA-2048 runs three times faster than ECC-256. It is considered that RSA-2048 runs 96,000 times faster than Weil pairing with 2 embedding degree, and 48,000 times faster than Weil pairing with 2 embedding degree and a fixed point.

2 The runtime for RSA-2048

The below number RSA-2048 has 617 digits, of 2048 bits.

$$
\begin{aligned}
n =\, & 22701801293785014193580405120204586741061235962766583907094021879215171483119139 \\
& 89487013309111104490168340094948384681829951804176350794892259077492546608817187 \\
& 92594659210265970467004498198990968620394600177430944738110569912941285428918808 \\
& 55362707407670722593737772666973440977361243336397308051763091506836310795312607 \\
& 23952036529003210584883950798145230729941718571579629745499502350531604091985919 \\
& 37180233074148804462179228008317660409386563445710347785534571210805307363945359 \\
& 23932651866030515041060966437313323672831539323500067937107541955437362433248361 \\
& 242525945868802353916766181532375855504886901432221349733
\end{aligned}
$$

Take $m = \text{IntegerPart}[n/2]$, and $k = n - 2$ (in the worst case), to compute $m^k \bmod n$. The Mathematica code for this computation is very simple.

```
 Timing[PowerMod[m, k, n]]
{0.015625,
2820045544329359416541292678855352434096021453838664407449354924101349
6891552195542526418848844207652306446485291873472009761833378678166506
9859253406497605101138041950407431489313766204811422795250340460515291
4358589540806744002922595758305289236172082622012724503982605450913700
5278982232672413459054235344040761394903449264044557562115788571320492
6289025448023591243317811369853477190934249524242065138808094154689438
9879164657634767671419474684366395302006312343403502916065231242410016
3291346124724638832633199137670965502839363870515376165952808914133596
06039708480086631216526684030920271126152864800801775651}
```

It spends about 0.015625 seconds, including only CPU time spent in the evaluation (AMD A9-9820 Processor 2.35 GHz, Mathematica11.0).

3 The runtime for ECC-256

We take the elliptic curve used for Bitcoin system,

$$y^2 = x^3 + 7 \bmod q \tag{1}$$

where $q = 115792089237316195423570985008687907853269984665640564039457584007908834671663$, a 256-bit prime, with a base point (a, b), where

$$a = 55066263022277343669578718895168534326250603453777594175500187360389116729240,$$
$$b = 32670510020758816978083085130507043184471273380659243275938904335757337482424$$

The arithmetic for the elliptic curve E/F_q is defined as follows. Given a point $P = (x, y)$ over the curve, its negative is $-P = (x, -y)$. For two points $P = (x_1, y_1), Q = (x_2, y_2), P \neq \pm Q$, the point addition is represented by $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

The point doubling is represented by $2(x_1, y_1) = (x_3, y_3)$ where

$$x_3 = \left(\frac{3x_1^2}{2y_1}\right)^2 - 2x_1, \quad y_3 = \left(\frac{3x_1^2}{2y_1}\right)(x_1 - x_3) - y_1$$

The Hasse's theorem gives an estimate of the number of points over E/F_q, $|\sharp(E/F_q) - (q+1)| \leq 2\sqrt{q}$.

```
AddPoint[point1_, point2_] :=  Module[{x1, y1, x2, y2, k, x3, y3, u, newpoint},
   x1 = point1[[1]]; y1 = point1[[2]]; x2 = point2[[1]]; y2 = point2[[2]];
   k = PowerMod[x2 - x1, -1, q]; u = Mod[(y2 - y1)*k, q];
   x3 = Mod[u^2 - x1 - x2, q]; y3 = Mod[u*(x1 - x3) - y1, q];
   newpoint = {x3, y3}]
DoublePoint[point_] := Module[{x1, y1, k, x3, y3, u, newpoint},
   x1 = point[[1]]; y1 = point[[2]];
   k = PowerMod[2*y1, -1, q]; u = Mod[3*x1^2*k, q];
   x3 = Mod[u^2 - 2*x1, q]; y3 = Mod[u*(x1 - x3) - y1, q];
   newpoint = {x3, y3}]
MultiPoint[k_, P_] := Module[{newpoint, BinaryTable, len, i, endpoint},
  BinaryTable = IntegerDigits[k, 2]; len = Length[BinaryTable]; newpoint = P;
  For[i = 2, i <= len, i++, If[BinaryTable[[i]] == 1,
    newpoint = AddPoint[DoublePoint[newpoint], P],
    newpoint = DoublePoint[newpoint]]]; endpoint = newpoint]
```

We now take $P = (a, b)$ and $k = q - 2$ (in the worst case) to compute kP.

```
a = 55066263022277343669578718895168534326250603453777594175500187360389116729240;
b = 32670510020758816978083085130507043184471273380659243275938904335757337482424;
P = {a, b}; k = q-2; Timing[MultiPoint[k, P]]
{0.046875,
{75937977013773973004625515363589527909731280618927128174417699995992069380903,
 21414141152327097618374269872214617344577357451074407808255142961578394379337}}
```

4 The runtime for ECC over a quadratic extended field

The polynomial $X^2 + 1$ is irreducible over F_q. The arithmetic for the elliptic curve

$$y^2 = x^3 + 7 \bmod (X^2 + 1, q) \tag{2}$$

has the same formulas as that over the curve $y^2 = x^3 + 7 \bmod q$, except the modulus $X^2 + 1$.

```
basePoly = X^2 + 1; moduliSet = {basePoly, q};
AddPoint1[point1_, point2_]:=Module[{x1, y1, x2, y2, k, x3, y3, u, newpoint},
   x1 = point1[[1]]; y1 = point1[[2]]; x2 = point2[[1]]; y2 = point2[[2]];
   k = PolynomialExtendedGCD[x2 - x1, basePoly, X, Modulus -> q][[2]][[1]];
   u = PolynomialMod[(y2 - y1)*k, moduliSet];
   x3 = PolynomialMod[u^2 - x1 - x2, moduliSet];
   y3 = PolynomialMod[u*(x1 - x3) - y1, moduliSet];
   newpoint = {x3, y3}];
DoublePoint1[point_]:=Module[{x1, y1, k, x3, y3, u, newpoint},
   x1 = point[[1]]; y1 = point[[2]];
   k = PolynomialExtendedGCD[2*y1, basePoly, X, Modulus -> q][[2]][[1]];
   u = PolynomialMod[3*x1^2*k, moduliSet];
   x3 = PolynomialMod[u^2 - 2*x1, moduliSet];
   y3 = PolynomialMod[u*(x1 - x3) - y1, moduliSet];
   newpoint = {x3, y3}];
MultiPoint1[k_, Q_]:= Module[{newpoint, BinaryTable, len, i, endpoint},
  BinaryTable = IntegerDigits[k, 2]; len = Length[BinaryTable];  newpoint = Q;
  For[i = 2, i <= len, i++, If[BinaryTable[[i]] == 1,
    newpoint = AddPoint1[DoublePoint1[newpoint], Q],
    newpoint = DoublePoint1[newpoint]]]; endpoint = newpoint]
```

To find a nontrivial point over the new curve, we suppose $(t, sX) \in E/F_{q^2}$,

$$s^2 X^2 = t^3 + 7 \bmod (X^2 + 1, q)$$

i.e., $s^2 = -t^3 - 7 \bmod q$. For t from 1 to 100, check if the right side is a quadratic residue modulo q. We then obtain a point

$$Q = (5, 23991821008281484097053715379747718372991279943638939452345024967188278261434X)$$

Take $k = \text{IntegerPart}[q^2/2]$ (in the worst case) to compute kQ.

```
Q={5, 23991821008281484097053715379747718372991279943638939452345024967188278261434*X};
k = IntegerPart[q^2/2]; Timing[MultiPoint1[k, Q]]
{1.46875,
{110415811740245324710223894840742945524568311653795374069402709165587086311706,
 107609321765704947133933396262394304713896020279760624970824647138603053168643 X}}
```

5 The runtime for ECC with the characteristic 2

The polynomial $X^{256} + X + 1$ is irreducible over F_2, which can be used to construct the extended field $F_{2^{256}}$. Let

$$y^2 + xy = x^3 + ax^2 + b \bmod (X^{256} + X + 1, 2) \tag{3}$$

be the elliptic curve, and $P = (x, y)$ be a point over the curve. Its negative is defined as $-P = (x, x + y)$. The point-addition is defined by: $(x_1, y_1) \neq \pm(x_2, y_2), (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, \quad y_3 = \frac{y_1 + y_2}{x_1 + x_2}(x_1 + x_3) + x_3 + y_1.$$

The point-doubling is defined by: $2(x_1, y_1) = (x_3, y_3)$, where

$$x_3 = \left(x_1 + \frac{y_1}{x_1}\right)^2 + \left(x_1 + \frac{y_1}{x_1}\right) + a, \quad y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3.$$

Take $a = X, b = 0$ and a base point $Q = (X^2, X^3)$.

```
basePoly = X^256 + X + 1; moduliSet = {basePoly, 2};
AddPoint2[point1_, point2_]:= Module[{x1, y1, x2, y2, k, x3, y3, u, newpoint},
   x1 = point1[[1]]; y1 = point1[[2]]; x2 = point2[[1]]; y2 = point2[[2]];
   k = PolynomialExtendedGCD[x1 + x2, basePoly, X, Modulus -> 2][[2]][[1]];
   u = PolynomialMod[(y1 + y2)*k, moduliSet];
   x3 = PolynomialMod[u^2 + u + x1 + x2 + X, moduliSet];
   y3 = PolynomialMod[u*(x1 + x3) + x3 + y1, moduliSet];
   newpoint = {x3, y3}];
DoublePoint2[point_]:= Module[{x1, y1, k, x3, y3, u, newpoint},
   x1 = point[[1]]; y1 = point[[2]];
   k = PolynomialExtendedGCD[x1, basePoly, X, Modulus -> 2][[2]][[1]];
   u = PolynomialMod[x1 + y1*k, moduliSet];
   x3 = PolynomialMod[u^2 + u + X, moduliSet];
   y3 = PolynomialMod[x1^2 + u*x3 + x3, moduliSet];
   newpoint = {x3, y3}];
MultiPoint2[k_, Q_] := Module[{newpoint, BinaryTable, len, i, endpoint},
  BinaryTable = IntegerDigits[k, 2]; len = Length[BinaryTable]; newpoint = Q;
  For[i = 2, i <= len, i++,
     If[BinaryTable[[i]] == 1,
        newpoint = AddPoint2[DoublePoint2[newpoint], Q],
        newpoint = DoublePoint2[newpoint]]];
  endpoint = newpoint]

k = 12345678909876556448897651344564432101130035144475884570079010980086640042002;
Q = {X^2, X^3}; Timing[MultiPoint2[k, Q]]

{7.10938, {1 + X^4 + X^9 + X^10 + X^12 + X^15 + X^19 + X^20 + X^21 +
   X^22 + X^24 + X^27 + X^28 + X^30 + X^32 + X^35 + X^37 + X^39 +
```

```
X^41 + X^45 + X^46 + X^47 + X^48 + X^49 + X^50 + X^54 + X^56 +
X^57 + X^62 + X^63 + X^65 + X^66 + X^67 + X^70 + X^72 + X^75 +
X^80 + X^83 + X^84 + X^85 + X^92 + X^95 + X^98 + X^99 + X^100 +
X^101 + X^104 + X^105 + X^106 + X^107 + X^108 + X^109 + X^110 +
X^111 + X^115 + X^116 + X^117 + X^118 + X^119 + X^121 + X^122 +
X^123 + X^125 + X^126 + X^127 + X^129 + X^130 + X^131 + X^132 +
X^134 + X^137 + X^139 + X^140 + X^142 + X^143 + X^144 + X^145 +
X^147 + X^148 + X^149 + X^151 + X^153 + X^154 + X^156 + X^159 +
X^161 + X^162 + X^167 + X^168 + X^169 + X^171 + X^172 + X^174 +
X^176 + X^181 + X^183 + X^186 + X^187 + X^188 + X^189 + X^193 +
X^195 + X^197 + X^199 + X^200 + X^203 + X^206 + X^207 + X^209 +
X^211 + X^212 + X^216 + X^217 + X^219 + X^222 + X^223 + X^224 +
X^225 + X^228 + X^232 + X^233 + X^234 + X^235 + X^236 + X^237 +
X^240 + X^243 + X^245 + X^246 + X^248 + X^249 + X^250 + X^252 +
X^254, 1 + X + X^4 + X^5 + X^7 + X^9 + X^10 + X^11 + X^12 + X^14 +
X^15 + X^16 + X^17 + X^18 + X^19 + X^22 + X^24 + X^26 + X^27 +
X^28 + X^31 + X^32 + X^33 + X^34 + X^37 + X^38 + X^39 + X^40 +
X^41 + X^43 + X^45 + X^48 + X^51 + X^52 + X^54 + X^55 + X^56 +
X^57 + X^60 + X^64 + X^65 + X^67 + X^69 + X^70 + X^71 + X^74 +
X^75 + X^77 + X^80 + X^81 + X^84 + X^86 + X^87 + X^89 + X^90 +
X^91 + X^92 + X^94 + X^95 + X^96 + X^97 + X^98 + X^99 + X^101 +
X^102 + X^107 + X^108 + X^109 + X^110 + X^112 + X^113 + X^114 +
X^125 + X^128 + X^130 + X^131 + X^133 + X^134 + X^135 + X^138 +
X^140 + X^141 + X^143 + X^144 + X^147 + X^148 + X^150 + X^152 +
X^156 + X^157 + X^158 + X^164 + X^166 + X^172 + X^173 + X^174 +
X^175 + X^177 + X^179 + X^181 + X^184 + X^186 + X^187 + X^188 +
X^189 + X^190 + X^191 + X^192 + X^202 + X^206 + X^208 + X^209 +
X^210 + X^211 + X^212 + X^213 + X^214 + X^215 + X^221 + X^223 +
X^224 + X^225 + X^228 + X^231 + X^236 + X^237 + X^238 + X^240 +
X^242 + X^245 + X^249 + X^250 + X^251 + X^252}}
```

6 The estimated runtime for pairings

6.1 Weil pairing

Let E be an elliptic curve over K, $p = \mathrm{char}(K)$, the integer $m \geqslant 2$, and $(m, p) = 1$. Then $\sum n_i(P_i)$ is a divisor of some function if and only if $\sum n_i = 0$ and $\sum [n_i]P_i = 0$, where

$$[n_i]P_i := \underbrace{P_i + P_i + \cdots + P_i}_{n_i \text{ times}}$$

Let $E[m] = \{P \in E : [m]P = O\}$. Then $\sharp E[m] = m^2$. If $d \,|\, m$, then $\sharp E[d] = d^2$. Hence, $E[m]$ can be expressed as $\mathbb{Z}_m \times \mathbb{Z}_m$, where $\mathbb{Z}_m = \{0, 1, \cdots, m-1\}$. If $T \in E[m]$, there exists $f \in \overline{K}(E)$ such that

$\mathrm{div}(f) = m(T) - m(O)$. Let $T' \in E$ and $[m]T' = T$. Then there exists $g \in \overline{K}(E)$ such that

$$\mathrm{div}(g) = \sum_{R \in E[m]} (T' + R) - (R),$$

which means that the composite functions $f \circ [m]$ and g^m have the same divisor. Hence, we assume that $f \circ [m] = g^m$. If $S \in E[m]$, for $\forall X \in E$, $g(X+S)^m = f([m]X + [m]S) = f([m]X) = g(X)^m$.

Let μ_m be the set of all m-th unit roots. The Weil-pairing is defined as [10]

$$\hat{e}_m : E[m] \times E[m] \to \mu_m, \quad \hat{e}_m(S, T) = g(X+S)/g(X)$$

where $X \in E$ is randomly picked such that $g(X+S) \neq 0, g(X) \neq 0$.

The logical dependency of involved functions and parameters in the definition is depicted by

$$T \xrightarrow{T=[m]T'} T' \xrightarrow{\mathrm{div}(g)=\sum_{R\in E[m]}(T'+R)-(R)} g.$$

Since $m^2 \,|\, \sharp E$, it seems impossible to compute T' from T. Actually, it is better to select T' first and then compute T. That is, the point T in the definition of Weil pairing should be fixed. In view of the importance of T, we replace the original notation with $\hat{e}_{(m;T)}$.

The map $\hat{e}_{(m;\cdot)}$ is:

- bilinear,

$$\hat{e}_{(m;T)}(S_1 + S_2,\, T) = \hat{e}_{(m;T)}(S_1,\, T)\hat{e}_{(m;T)}(S_2,\, T),$$
$$\hat{e}_{(m;T_1+T_2)}(S,\, T_1 + T_2) = \hat{e}_{(m;T_1)}(S,\, T_1)\hat{e}_{(m;T_2)}(S,\, T_2);$$

- alternative, $\hat{e}_{(m;T)}(S,\, T) = \hat{e}_{(m;S)}(T,\, S)^{-1}$;
- non-degenerate, if $\forall\, S \in E[m]$, $\hat{e}_{(m;T)}(S,\, T) = 1$ holds, then $T = O$.

In fact, by the definition of $\hat{e}_{(m;\cdot)}$ and the randomness of X, we have

$$\hat{e}_{(m;T)}(S_1 + S_2,\, T) = \frac{g(X + S_1 + S_2)}{g(X + S_1)} \frac{g(X + S_1)}{g(X)} = \hat{e}_{(m;T)}(S_1,\, T)\hat{e}_{(m;T)}(S_2,\, T).$$

Let $f_1, f_2, f_3, g_1, g_2, g_3$ be the functions corresponding to $T_1, T_2, T_3 = T_1 + T_2$, in the definition of Weil pairing. Select $h \in \overline{K}(E)$ such that $\mathrm{div}(h) = (T_1 + T_2) - (T_1) - (T_2) + (O)$. Hence, $\mathrm{div}(f_3/f_1 f_2) = m\,\mathrm{div}(h)$, i.e., there is $c \in \overline{K}^*$ such that $f_3 = c f_1 f_2 h^m$. Since $f_i \circ [m] = g_i^m$, there is $c' \in \overline{K}^*$ such that $g_3 = c' g_1 g_2 (h \circ [m])$. Therefore,

$$\begin{aligned}\hat{e}_{(m;T_1+T_2)}(S,\, T_1 + T_2) =& \frac{g_3(X+S)}{g_3(X)} = \frac{g_1(X+S)g_2(X+S)h([m]X + [m]S)}{g_1(X)g_2(X)h([m]X)} \\ =& \hat{e}_{(m;T_1)}(S,\, T_1)\hat{e}_{(m;T_2)}(S,\, T_2).\end{aligned}$$

Strictly speaking, the above is not linear because the equation contains three different maps. Henceforth, we still habitually call $\hat{e}_{(m;T)}$ a bilinear map.

6.2 Miller algorithm

As wee see, the definition of Weil pairing depends on the selection of function g, but it is difficult to find g directly. We now introduce other equivalent forms of Weil pairing. Let C be a smooth elliptic curve. $D = \sum n_p(P) \in Div(C)$, $f \in \overline{K}(C)^*$, $\mathrm{supp}(\mathrm{div}(f)) \cap \mathrm{supp}(D) = \emptyset$, where $\mathrm{supp}(D)$ denotes the support of D, which is the set consists of the points with non-zero multiplicity. Define

$$f(D) = \prod_{P \in C} f(P)^{n_p}.$$

Suppose the integer $n > 1$, and D_1, D_2 are two divisors of C such that $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$. Pick two functions f_1, f_2 such that $\mathrm{div}(f_i) = nD_i$, $i = 1, 2$, and define the Weil pairing as $\hat{e}_n(D_1, D_2) = f_1(D_2)/f_2(D_1)$.

Let $P, Q \in E[n]$. Select $T \in E$ and $D_1 = ([P+T] - [T]), D_2 = ([Q] - [O])$ such that $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$. The Weil pairing can also be defined as

$$\hat{e}_n(P, Q) := \hat{e}_n([P+T] - [T], [Q] - [O]).$$

Now it suffices to find two functions f_1, f_2 such that $\mathrm{div}(f_1) = n([P+T]-[T])$, $\mathrm{div}(f_2) = n([Q]-[O])$. The Miller algorithm can be used to find such functions.

Let E be an elliptic curve, $P, Q \in E[n]$. Denote the line through two points P, Q by $L_{P,Q} = 0$. If $P = Q$, $L_{P,P} = 0$ is defined as the tangent line through the point P. Hence, $\mathrm{div}(L_{P,Q}) = [P] + [Q] + [-(P+Q)] - 3[O]$. Define

$$h_{P,Q} = \frac{L_{P,Q}}{L_{P+Q,-(P+Q)}}.$$

Clearly, $\mathrm{div}(h_{P,Q}) = [P] + [Q] - [P+Q] - [O]$.

Let $P \in E$, $f_{0,P} = f_{1,P} = 1$. For a positive integer n, define $f_{n+1,P} := f_{n,P}\, h_{p,nP}$. Then $\mathrm{div}(f_{n,p}) = n[P] - (n-1)[O] - [nP]$. If $nP = O$, then $\mathrm{div}(f_{n,p}) = n[P] - n[O]$.

A direct computation for $f_{n,P}$ based on the above recurrence relation is infeasible, if n is very large. In practice, it is better to use the so-called addition chain to compute $f_{n,P}$, due to that

$$f_{m+n,P} = f_{m,P} \cdot f_{n,P} \cdot h_{mP,nP}$$

Since two rational functions with the same divisor are identical except for a constant factor, it only needs to check that the both sides of the equation have a same divisor.

It is easy to find that f_1 and $f_{n,P}$ are very similar except a shift transformation. Hence, we have $\mathrm{div} f_1 = \mathrm{div}(f_{n,P} \circ \lambda_{-T})$, where $\lambda_{-T} : P \to P - T$. At this point, we have completed the construction of f_1. Likewise, we can construct f_2 such that $\mathrm{div} f_2 = \mathrm{div}(f_{n,Q})$. We now have the more concise representation of Weil pairing [7],

$$\begin{aligned} \hat{e}_n(P, Q) = \hat{e}_n([P+T] - [T], [Q] - [O]) &= \frac{f_1([Q] - [O])}{f_2([P+T] - [T])} \\ &= \frac{f_1(Q)}{f_1(O)} \frac{f_2(T)}{f_2(P+T)} = \frac{f_{n,P}(Q-T)}{f_{n,P}(-T)} \frac{f_{n,Q}(T)}{f_{n,Q}(P+T)}. \end{aligned}$$

Taking $T \to O$, it gives

$$\hat{e}_n(P,Q) = (-1)^n \frac{f_{n,P}(Q)}{f_{n,Q}(P)} \tag{4}$$

For $\forall P \in E[n]$, $\hat{e}_n(P,P) = \pm 1$. So, the definition should be revised by using a homomorphic map.

The map $\hat{e}_n$ is defined over an n-torsion group. For its existence, we have the following result [1].

Let E be an elliptic curve over the field $\mathbb{F}_q$, n be a prime and $n \mid \sharp E(\mathbb{F}_q)$. If $\gcd(n,q) = 1$, $n \nmid q-1$, then $E[n] \subset E(\mathbb{F}_{q^k})$ if and only if $n \mid q^k - 1$. In this case, the group μ_n of all n-th unit roots satisfies that

$$\mu_n \subset \mathbb{F}_{q^k}, \quad \mu_n \not\subset \mathbb{F}_{q^j}, j = 1, \cdots, k-1$$

where k is called the embedding degree of $E[n]$ with respect to $E(\mathbb{F}_{q^k})$. The result indicates that the computation of $\hat{e}_n$ is always done over the field $\mathbb{F}_{q^k}$. From the practical point of view, it is usual to specify that $k \leqslant 6$ in order to facilitate the computation of pairings.

We now take the embedding degree $k = 2$, and

$$\begin{aligned}
&q = 115792089237316195423570985008687907853269984665640564039457584007908834671663,\\
&Q = (5, 23991821008281484097053715379747718372991279943638939452345024967188278261434X)
\end{aligned}$$

Suppose the order n is of the binary string $b_t b_{t-1} \cdots b_1 b_0$. Let

$$n_k = n - (b_0 + 2b_1 + \cdots + 2^k b_k), 0 \le k \le t,$$

i.e., $b_0 + n_0 = n, (b_0 + 2b_1) + n_1 = n, (b_0 + 2b_1 + 2^2 b_2) + n_2 = n, \cdots$. We then have

$$\begin{aligned}
f_{n,Q} &= f_{n_0,Q} \cdot f_{b_0,Q} \cdot h_{n_0Q,b_0Q} = f_{n_0,Q} \cdot f_{b_0,Q} \cdot \frac{L_{n_0Q,b_0Q}}{L_{nQ,-nQ}} \\
&= f_{n_1,Q} \cdot f_{2b_1,Q} \cdot h_{n_1Q,2b_1Q} \cdot f_{b_0,Q} \cdot \frac{L_{n_0Q,b_0Q}}{L_{nQ,-nQ}} \\
&= f_{n_1,Q} \cdot f_{2b_1,Q} \cdot f_{b_0,Q} \cdot \frac{L_{n_0Q,b_0Q}}{L_{nQ,-nQ}} \cdot \frac{L_{n_1Q,2b_1Q}}{L_{n_0Q,-n_0Q}} \\
&= f_{n_2,Q} \cdot f_{2^2b_2,Q} \cdot h_{n_2Q,2^2b_2Q} \cdot f_{2b_1,Q} \cdot f_{b_0,Q} \cdot \frac{L_{n_0Q,b_0Q}}{L_{nQ,-nQ}} \cdot \frac{L_{n_1Q,2b_1Q}}{L_{n_0Q,-n_0Q}} \\
&= f_{n_2,Q} \cdot f_{2^2b_2,Q} \cdot f_{2b_1,Q} \cdot f_{b_0,Q} \cdot \frac{L_{n_0Q,b_0Q}}{L_{nQ,-nQ}} \cdot \frac{L_{n_1Q,2b_1Q}}{L_{n_0Q,-n_0Q}} \cdot \frac{L_{n_2Q,2^2b_2Q}}{L_{n_1Q,-n_1Q}} \\
&= \cdots \\
&= f_{2^tb_t,Q} \cdots f_{2^2b_2,Q} \cdot f_{2b_1,Q} \cdot f_{b_0,Q} \cdot \frac{L_{n_0Q,b_0Q}}{L_{nQ,-nQ}} \cdot \frac{L_{n_1Q,2b_1Q}}{L_{n_0Q,-n_0Q}} \cdot \frac{L_{n_2Q,2^2b_2Q}}{L_{n_1Q,-n_1Q}} \cdots \frac{L_{(n-n_t)Q,2^tb_tQ}}{L_{n_{t-1}Q,-n_{t-1}Q}}
\end{aligned}$$

In this process, we need to compute the points

$$b_0Q, 2b_1Q, 2^2b_2Q, \cdots, 2^tb_tQ;\ n_0Q, n_1Q, n_2Q, \cdots, n_{t-1}Q$$

Likewise, for the other point P, we need to compute the points

$$b_0P, 2b_1P, 2^2b_2P, \cdots, 2^tb_tP;\ n_0P, n_1P, n_2P, \cdots, n_{t-1}P$$

The cost for evaluating the pairing Eq.(4), is almost $2t$ times that of computing kP over E/F_{q^2}. Practically, $t \approx 2 \times 256$, and $2t \approx 1024$.

We fail to generate the Mathematica code for testing Weil pairings, due to the hardness to compute the order of point Q over the curve $y^2 = x^3 + 7 \bmod (X^2 + 1, q)$.

7 The runtime comparison

Pairing-based cryptography should specify that the base point $P \in E/F_{q^k}$ is of a large order so that ECDLP must be intractable. Besides, it should specify that the order of μ_n should be large enough so that the general discrete logarithm must also be intractable. The practical runtimes for RSA-2048, ECC-256, ECC over an extended field, and the estimated runtimes for Weil pairings are listed below (see Table 2).

Table 2: The comparison for different runtimes in the worst cases

RSA-2048	0.015625 (seconds)
ECC over F_q	0.046875
ECC over F_{q^2}	1.46875
ECC over $F_{2^{256}}$	7.10938
Weil pairing over F_{q^2}	1024×1.46875
Weil pairing over F_{q^2} with a fixed point	512×1.46875

The runtime for RSA-2048 is almost three times faster than ECC over F_q, 450 times faster than ECC over $F_{2^{256}}$, 96,000 times faster than Weil pairing with 2 embedding degree, and 48,000 times faster than Weil pairing with 2 embedding degree and a fixed point.

8 Conclusion

RSA has survived over forty years due to its straightforward principle and fast performance. In view of the current unconvincing quantum machines, we anticipate the coexistence of RSA and ECC will last at least ten years.

References

[1] R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has sub-exponential discrete log problem under the menezes-okamoto-vanstone algorithm. *Journal of Cryptology*, (11):141–145, 1998.

[2] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[3] D. Castelvecchi. Ibm releases first-ever 1,000-qubit quantum chip. *Nature*, 624(238), 2023.

[4] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, USA, 2003.

[5] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[6] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer.

[7] V. Miller. The weil pairing, and its efficient calculation. *Journal of Cryptology*, (17):235–261, 2004.

[8] R. Rivest, A. Shamir, and L. Adleman. A mehtod for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[9] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[10] J. Silverman. *The arithmetic of elliptic curves*. Springer, 1986.