Matrix Code Equivalence

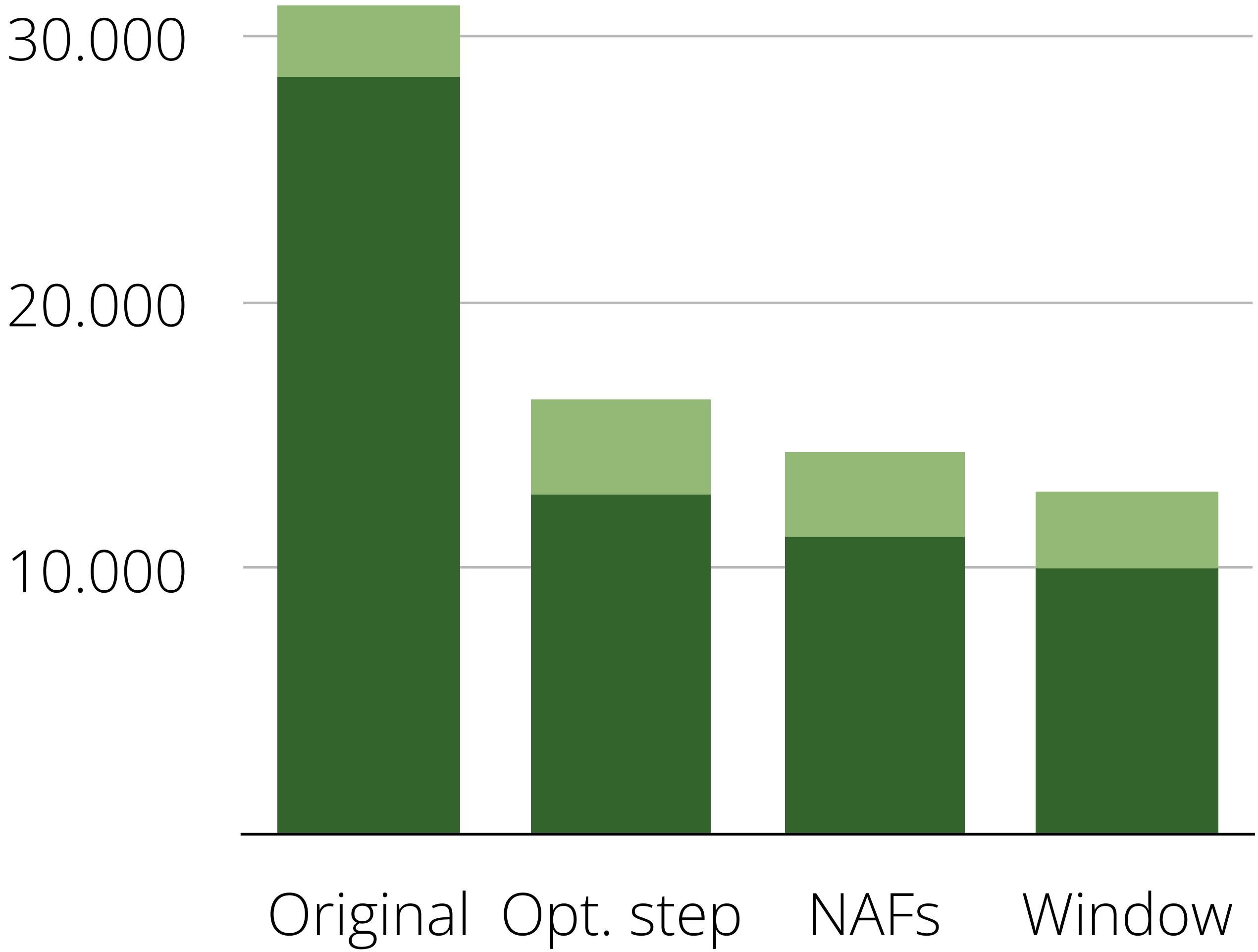
1

Radboud University

IN DEI NOMINE FELICITER

Speeding-up general pairings

2


30.000
20.000
10.000
Original
Opt. step
NAFs
Window

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

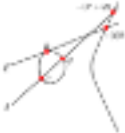


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\mathrm{prod}} = \ell\ell'$ is $(\ell_{\mathrm{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\mathrm{quo}} = \ell/\ell'$ is $(\ell_{\mathrm{quo}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{Y-\lambda_1 X-\nu_1 Z}{Y-\lambda_2 X-\nu_2 Z}$ gives $\frac{1-\lambda_1 X/Y-\nu_1 Z/Y}{1-\lambda_2 X/Y-\nu_2 Z/Y}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y-\lambda_1 x-\nu_1)(y-\lambda_2 x-\nu_2)$, substituted the y^2 for x^3+ax+b and wrote $y = \frac{x^3+ax+b+(\lambda_1 x+\nu_1)(\lambda_2 x+\nu_2)-\ell\ell'}{(\lambda_1+\lambda_2)x+\nu_1+\nu_2}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3+ax+b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These large costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,Q'}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
1:  $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
2:  for $i$ from 0 to $l - 1$ do
3:      if ($m_i = 1$) then
4:          $A_R[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j + 1$
5:      end if
6:      $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
7:  end for
8:  $R' \leftarrow A_R[0]$,   $f \leftarrow A_f[0]$
9:  for ($j \leftarrow 1$; $j \le \mathrm{hw}(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,   $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P x_{i,P} \left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)} (x_Q - x_{i,P}) + 1 \right).$$

We write this as $\theta y_Q + y_P \delta_i$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P \alpha_{2i,P} + \theta y_Q \beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2 \gamma_i - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]\delta/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = (3x_P^2 + Ax_P + B)(\theta_i \beta_{i,P} + \alpha_{i,P}^2)/(2x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P \left(\frac{y_{iP} - 1}{x_{iP} - x_P} (x - x_P) - 1 \right).$$

and so

$$l(Q) = \theta y_Q - y_P \left(\frac{y_{iP} - 1}{x_{iP} - x_P} (x_Q - x_P) + 1 \right).$$

Writing this as $\theta y_Q + y_P \delta_i$, we have $f_{(i+1),P} = \alpha_{(i+1),P} + \theta y_P y_Q \beta_{(i+1),P}$ where

$$\alpha_{(i+1)} = (3x_P^2 + Ax_P + B)\alpha_{i,P}\delta_i - (x_Q^3 + Ax_Q + B)\beta_{i,P}/(2x_Q - x_{(i+1),P});$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} + \beta_{i,P}\delta_i)/(x_Q - x_{(i+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

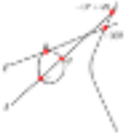


Figure 3.5: The functions ℓ and ℓ' on E.

the function $f_{\text{prod}} = \ell\ell'$ is $(f_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $f_{\text{quot}} = \ell/\ell'$ is $(f_{\text{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that f_{quot} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{Y - \lambda_1 X - \nu_1 Z}{Y - \lambda_2 X - \nu_2 Z}$ gives $\frac{1 - \lambda_1 X/Y - \nu_1 Z/Y}{1 - \lambda_2 X/Y - \nu_2 Z/Y}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{\ell\ell' - (x^3 + ax + b) - \ldots}{\ldots}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on $\mathbb{K}^2$) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These large costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G_2'$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell$
 1: $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
 2: for i from 0 to l − 1 do
 3:     if ($m_i = 1$) then
 4:         $A_{R'}[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j + 1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_{R'}[0]$,   $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le hw(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,   $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P \alpha_{k,P}\left(\frac{3x_{k,P}^2 + A}{2(Y_{k,P}^2 + x_{k,P} + B)}(x_Q - x_{k,P}) + 1\right).$$

We write this as $\theta y_Q + y_P \lambda_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2k,P}$.

Multiplying all these together gives $f_{2k,P} = y_P \alpha_{2k,P} + \theta y_Q \beta_{2k,P}$ where

$$\alpha_{2k,P} = (Y_{k,P}^2 \lambda_1 - (x_Q^3 + Ax_Q + B)(Y_{k,P}^2))/(x_Q - x_{2k,P})$$

and

$$\beta_{2k,P} = ((x_P^3 + Ax_P + B)(\lambda_1 Y_{k,P}^2 + \alpha_{k,P}^2))/(x_Q - x_{2k,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{\alpha_{kP} - 1}{x_{kP} - x_P}(x - x_P) + 1\right),$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{\alpha_{k,P} - 1}{x_{kP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P \lambda_2$, we have $f_{(k+1),P} = \alpha_{(k+1),P} + \theta y_P y_Q \beta_{(k+1),P}$ where

$$\alpha_{(k+1)} = ((x_P^3 + Ax_P + B)\alpha_{k,P}\lambda_2 - (x_Q^3 + Ax_Q + B)\beta_{k,P})/(x_Q - x_{(k+1),P})$$

and

$$\beta_{(k+1),P} = (\alpha_{k,P} - \beta_{k,P}\lambda_2)/(x_Q - x_{(k+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-2R) - 3(\mathcal{O})$. The divisor of

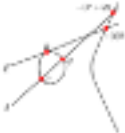


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{prod} = \ell\ell'$ is $(\ell_{prod}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P + Q)) + (-2R) - 6(\mathcal{O})$. The divisor of $\ell_{quo} = \ell/\ell'$ is $(\ell_{quo}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-2R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y-\lambda_1 x-\nu_1}{y-\lambda_2 x-\nu_2}$ gives $\frac{Y-\lambda_1 X-\nu_1 Z}{Y-\lambda_2 X-\nu_2 Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{[illegible]}{[illegible]}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G_2'$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
 1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
 2: for $i$ from 0 to $l - 1$ do
 3:     if ($m_i = 1$) then
 4:         $A_R[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j + 1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_R[0]$,  $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le \mathrm{wt}(m) - 1$; $j$++) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,  $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_i$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P\alpha_{2i,P} + \theta y_Q\beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha^2_{i,P}c_i - (x_Q^3 + Ax_Q + B)\beta^2_{i,P}]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_Q^3 + Ax_Q + B)\theta\beta_{i,P} + \alpha^2_{i,P})/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the values. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{u_{2i}-1}{x_{2i}-x_P}(x - x_P) + 1\right).$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{u_{2i}-1}{x_{2i}-x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P d_i$, we have $f_{(2i+1),P} = \alpha_{2i+1,P} + \theta y_P y_Q\beta_{2i+1,P}$ where

$$\alpha_{2i+1} = ((x_Q^3 + Ax_Q + B)\alpha_{2i,P}d_i - (x_Q^3 + Ax_Q + B)\beta_{2i,P})/(x_Q - x_{2i+1,P}),$$

and

$$\beta_{2i+1,P} = (\alpha_{2i,P} - \beta_{2i,P}d_i)/(x_Q - x_{2i+1,P}).$$

(Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda' x + \nu'$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

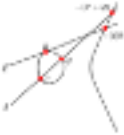


Figure 3.5: The functions ℓ and ℓ' on E.

the function $f_{\text{prod}} = \ell\ell'$ is $(f_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $f_{\text{quot}} = \ell/\ell'$ is $(f_{\text{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that f_{quot} does not intersect E at $\mathcal{O}$: projectifying $\ell/\ell' = \frac{Y - \lambda X - \nu Z}{Y - \lambda' X - \nu' Z}$ gives $\frac{Y/Z - \lambda X/Z - \nu}{Y/Z - \lambda' X/Z - \nu'}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda x - \nu)(y - \lambda' x - \nu')$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = $ [illegible]. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on $\mathbb{A}^2$) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G_2'$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \dots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
2: for $i$ from $0$ to $l-1$ do
3:     if ($m_i = 1$) then
4:         $A_{R'}[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j+1$
5:     end if
6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
7: end for
8: $R' \leftarrow A_{R'}[0]$,  $f \leftarrow A_f[0]$
9: for ($j \leftarrow 1$; $j \le \mathrm{hw}(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,  $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P c_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P\alpha_{2i,P} + \theta y_Q\beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2 c_1 - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_Q^3 + Ax_Q + B)\theta_Q\beta_{i,P}^2 + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{y_{2i} - 1}{x_{2i} - x_P}(x - x_P) + 1\right),$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{y_{2i} - 1}{x_{2i} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_2$, we have $f_{2i+1,P} = \alpha_{2i+1,P} + \theta y_P y_Q\beta_{2i+1,P}$ where

$$\alpha_{2i+1} = ((x_Q^3 + Ax_Q + B)\alpha_{2i,P}c_2 - (x_Q^3 + Ax_Q + B)\beta_{2i,P})/(x_Q - x_{2i+1,P})$$

and

$$\beta_{2i+1,P} = (\alpha_{2i,P} + \beta_{2i,P}c_2)/(x_Q - x_{2i+1,P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

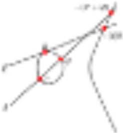


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\text{prod}} = \ell\ell'$ is $(\ell_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\text{quo}} = \ell/\ell'$ is $(\ell_{\text{quo}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$: projectifying $\ell/\ell' = \frac{Y - \lambda_1 X - \nu_1 Z}{Y - \lambda_2 X - \nu_2 Z}$ gives $\frac{1 - \lambda_1 X/Y - \nu_1 Z/Y}{1 - \lambda_2 X/Y - \nu_2 Z/Y}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{x^3 + ax + b + (\lambda_1 x + \nu_1)(\lambda_2 x + \nu_2)}{(\lambda_1 + \lambda_2)x + \nu_1 + \nu_2}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$
1: $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
2: for $i$ from 0 to $l-1$ do
3:     if ($m_i = 1$) then
4:         $A_{R'}[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j+1$
5:     end if
6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
7: end for
8: $R' \leftarrow A_{R'}[0]$,   $f \leftarrow A_f[0]$
9: for ($j \leftarrow 1$; $j \leq hw(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,   $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_{k,P}\left(\frac{3x_{k,P}^2 + A}{2(x_{k,P}^3 + Ax_{k,P} + B)}(x_Q - x_{k,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_k$. The vertical line contributes simply $v(Q) = x_Q - x_{2k,P}$.
Multiplying all these together gives $f_{2k,P} = y_P\alpha_{2k,P} + \theta y_Q\beta_{2k,P}$ where

$$\alpha_{2k,P} = [\alpha_{k,P}^2 c_k - (x_Q^3 + Ax_Q + B)\beta_{k,P}^2]/(x_Q - x_{2k,P})$$

and

$$\beta_{2k,P} = ((x_P^3 + Ax_P + B)c_k\beta_{k,P}^2 + \alpha_{k,P}^2)/(x_Q - x_{2k,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{y_k - 1}{x_k - x_P}(x - x_P) + 1\right),$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{y_{k,P} - 1}{x_k - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P d_k$, we have $f_{k+1,P} = \alpha_{k+1,P} + \theta y_P y_Q\beta_{k+1,P}$ where

$$\alpha_{k+1,P} = ((x_P^3 + Ax_P + B)\alpha_{k,P}d_k - (x_Q^3 + Ax_Q + B)\beta_{k,P})/(x_Q - x_{k+1,P})$$

and

$$\beta_{k+1,P} = (\alpha_{k,P} + \beta_{k,P}d_k)/(x_Q - x_{k+1,P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda' x + \nu'$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

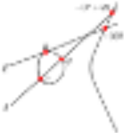


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\rm prod} = \ell\ell'$ is $(\ell_{\rm prod}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\rm quot} = \ell/\ell'$ is $(\ell_{\rm quot}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that $\ell_{\rm quot}$ does not intersect E at $\mathcal{O}$: projectifying $\ell/\ell' = \frac{y-\lambda x-\nu}{y-\lambda' x-\nu'}$ gives $\frac{Y-\lambda X-\nu Z}{Y-\lambda' X-\nu' Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda x - \nu)(y - \lambda' x - \nu')$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{[illegible]}{[illegible]}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on $\overline{K}^2$) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These large costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G_2'$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,Q'}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
 1: $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
 2: for $i$ from $0$ to $l-1$ do
 3:     if ($m_i = 1$) then
 4:         $A_R[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j+1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_R[0]$,   $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \leq hw(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,   $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(\bar{Q}) = \theta y_Q - y_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_i$. The vertical line contributes simply $v(\bar{Q}) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P \alpha_{2i,P} + \theta y_Q \beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2 c_i - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_P^3 + Ax_P + B)\beta_i \beta_{i,P} + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{\kappa_{i,P} - 1}{x_{iP} - x_P}(x - x_P) + 1\right).$$

and so

$$l(\bar{Q}) = \theta y_Q - y_P\left(\frac{\kappa_{i,P} - 1}{x_{iP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P d_i$, we have $f_{(i+1),P} = \alpha_{(i+1),P} + \theta y_P y_Q \beta_{(i+1),P}$ where

$$\alpha_{(i+1)} = ((x_P^3 + Ax_P + B)\alpha_{i,P} d_i - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P})$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} + \beta_{i,P} d_i)/(x_Q - x_{(i+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

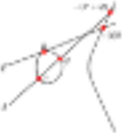


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\text{prod}} = \ell\ell'$ is $(\ell_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\text{quo}} = \ell/\ell'$ is $(\ell_{\text{quo}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y-\lambda_1 x-\nu_1}{y-\lambda_2 x-\nu_2}$ gives $\frac{Y-\lambda_1 X-\nu_1 Z}{Y-\lambda_2 X-\nu_2 Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y-\lambda_1 x-\nu_1)(y-\lambda_2 x-\nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y =$ [illegible]. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These large costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \dots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
 1: $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
 2: for i from 0 to l − 1 do
 3:     if ($m_i = 1$) then
 4:         $A_{R'}[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j + 1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_{R'}[0]$,   $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le \mathrm{wt}(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,   $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return f
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P \alpha_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P \alpha_{2i,P} + \theta y_Q \beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2 c_1 - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_P^3 + Ax_P + B)\theta_i \beta_{i,P} + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{\alpha_{iP} - 1}{x_{iP} - x_P}(x - x_P) + 1\right).$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{\alpha_{i,P} - 1}{x_{iP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_2$, we have $f_{(i+1),P} = \alpha_{i+1,P} + \theta y_P y_Q \beta_{i+1,P}$ where

$$\alpha_{(i+1)} = ((x_P^3 + Ax_P + B)\alpha_{i,P} c_2 - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P}),$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} - \beta_{i,P} c_2)/(x_Q - x_{(i+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-2R) - 3(\mathcal{O})$. The divisor of

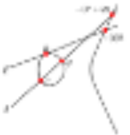


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{prod} = \ell\ell'$ is $(\ell_{prod}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-2R) - 6(\mathcal{O})$. The divisor of $\ell_{quo} = \ell/\ell'$ is $(\ell_{quo}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-2R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y - \lambda_1 x - \nu_1}{y - \lambda_2 x - \nu_2}$ gives $\frac{Y - \lambda_1 X - \nu_1 Z}{Y - \lambda_2 X - \nu_2 Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{[illegible]}{[illegible]}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \dots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,Q'}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
 1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
 2: for $i$ from 0 to $l-1$ do
 3:     if ($m_i = 1$) then
 4:         $A_{R'}[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j+1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_{R'}[0]$,  $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le hw(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,  $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta_{Q} - y_{iP}\left(\frac{3x_{i,P}^2 + A}{2(Y_{i,P}^2 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta_{Q} + y_Q c_i$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_Q\alpha_{2i,P} + \theta_{Q}\beta_{2i,P}$ where

$$\alpha_{2i,P} = (\alpha_{i,P}^2 c_i - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2)/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_P^3 + Ax_P + B)\theta_Q\beta_{i,P}^2 + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{y_{iP} - 1}{x_{iP} - x_P}(x - x_P) + 1\right),$$

and so

$$l(Q) = \theta_{Q} - y_P\left(\frac{y_{i,P} - 1}{x_{iP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta_{Q} + y_Q c_i$, we have $f_{(i+1),P} = \alpha_{(i+1),P} + \theta_Q y_Q\beta_{(i+1),P}$ where

$$\alpha_{(i+1)} = ((x_P^3 + Ax_P + B)\alpha_{i,P}c_i - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P}),$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} - \beta_{i,P}c_i)/(x_Q - x_{(i+1),P}).$$

$Q)$ with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

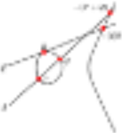


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\rm prod} = \ell\ell'$ is $(\ell_{\rm prod}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\rm quo} = \ell/\ell'$ is $(\ell_{\rm quo}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that $\ell_{\rm quo}$ does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y-\lambda_1 x-\nu_1}{y-\lambda_2 x-\nu_2}$ gives $\frac{Y-\lambda_1 X-\nu_1 Z}{Y-\lambda_2 X-\nu_2 Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y-\lambda_1 x-\nu_1)(y-\lambda_2 x-\nu_2)$, substituted the y^2 for x^3+ax+b and wrote $y = \frac{(x^3+ax+b)+(\lambda_1 x+\nu_1)(\lambda_2 x+\nu_2)}{(\lambda_1+\lambda_2)x+\nu_1+\nu_2}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on $\mathbb{A}^2$) behave at points that are not on E, where the substitution $y^2 = x^3+ax+b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These large costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,Q'}(P)$ representing a class in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$
 1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
 2: for $i$ from 0 to $l-1$ do
 3:     if ($m_i = 1$) then
 4:         $A_{R'}[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j+1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_{R'}[0]$,  $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le wt(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,  $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P\alpha_{i,P}\left(\frac{3x_{iP}^2 + A}{2(x_{iP}^3 + Ax_{iP} + B)}(x_Q - x_{iP}) + 1\right).$$

We write this as $\theta y_Q + y_P c_i$. The vertical line contributes simply $v(Q) = x_Q - x_{2iP}$.

Multiplying all these together gives $f_{2i,P} = y_P\alpha_{2i,P} + \theta y_Q\beta_{2i,P}$ where

$$\alpha_{2i,P} = [\gamma_{i,P}^2 c_i - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2iP})$$

and

$$\beta_{2i,P} = ((x_Q^3 + Ax_Q + B)\theta_i\beta_{i,P} + c_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{u_{i} - 1}{x_{iP} - x_P}(x - x_P) + 1\right).$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{u_{i} - 1}{x_{iP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_i$, we have $f_{(i+1),P} = \alpha_{i+1,P} + \theta y_P y_Q\beta_{i+1,P}$ where

$$\alpha_{i+1,P} = ((x_Q^3 + Ax_Q + B)\alpha_{i,P}c_i - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P})$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} - \beta_{i,P}c_i)/(x_Q - x_{(i+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

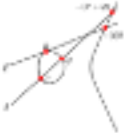


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\mathrm{prod}} = \ell\ell'$ is $(\ell_{\mathrm{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\mathrm{quot}} = \ell/\ell'$ is $(\ell_{\mathrm{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quot} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{Y - \lambda_1 X - \nu_1 Z}{Y - \lambda_2 X - \nu_2 Z}$ gives $\frac{1 - \lambda_1 X/Y - \nu_1 Z/Y}{1 - \lambda_2 X/Y - \nu_2 Z/Y}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{[illegible]}{[illegible]}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than as the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$
1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
2: for $i$ from $0$ to $l - 1$ do
3:     if ($m_i = 1$) then
4:         $A_{R'}[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j + 1$
5:     end if
6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
7: end for
8: $R' \leftarrow A_{R'}[0]$,  $f \leftarrow A_f[0]$
9: for ($j \leftarrow 1$; $j \le hw(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,  $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P y_{n,P}\left(\frac{3x_{n,P}^2 + A}{2(x_{n,P}^3 + Ax_{n,P} + B)}(x_Q - x_{n,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2n,P}$.

Multiplying all these together gives $f_{2n,P} = y_P \alpha_{2n,P} + \theta y_Q \beta_{2n,P}$ where

$$\alpha_{2n,P} = [\alpha_{n,P}^2 c_1 - (x_Q^3 + Ax_Q + B)\beta_{n,P}^2]/(x_Q - x_{2n,P})$$

and

$$\beta_{2n,P} = ((x_P^3 + Ax_P + B)\theta_1 \beta_{n,P}^2 + \alpha_{n,P}^2)/(x_Q - x_{2n,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{y_{n,P} - 1}{x_{n,P} - x_P}(x - x_P) + 1\right),$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{y_{n,P} - 1}{x_{n,P} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_2$, we have $f_{n+1,P} = \alpha_{n+1,P} + \theta y_Q \beta_{n+1,P}$ where

$$\alpha_{n+1,P} = ((x_P^3 + Ax_P + B)\alpha_{n,P} c_2 - (x_Q^3 + Ax_Q + B)\beta_{n,P})/(x_Q - x_{n+1,P})$$

and

$$\beta_{n+1,P} = (\alpha_{n,P} + \beta_{n,P} c_2)/(x_Q - x_{n+1,P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda' x + \nu'$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

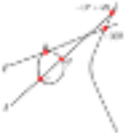


Figure 3.5: The functions ℓ and ℓ' on E.

the function $f_{\text{prod}} = \ell\ell'$ is $(f_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $f_{\text{quot}} = \ell/\ell'$ is $(f_{\text{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that f_{quot} does not intersect E at $\mathcal{O}$: projectifying $\ell/\ell' = \frac{y - \lambda x - \nu}{y - \lambda' x - \nu'}$ gives $\frac{Y - \lambda X - \nu Z}{Y - \lambda' X - \nu' Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda x - \nu)(y - \lambda' x - \nu')$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{(x^3+ax+b)+(\lambda x+\nu)(\lambda' x+\nu')}{(\lambda+\lambda')x+\nu+\nu'}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than for the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \dots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
1: $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
2: for $i$ from 0 to $l-1$ do
3:     if $(m_i = 1)$ then
4:         $A_R[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j+1$
5:     end if
6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
7: end for
8: $R' \leftarrow A_R[0]$,   $f \leftarrow A_f[0]$
9: for $(j \leftarrow 1;\ j \le h(m) - 1;\ j{+}{+})$ do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,   $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P c_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P \alpha_{2i,P} + \theta y_Q \beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2 c_1 - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_P^3 + Ax_P + B)\theta_i \beta_{i,P}^2 + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{c_{i,P} - 1}{x_{i,P} - x_P}(x - x_P) + 1\right),$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{c_{i,P} - 1}{x_{i,P} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_2$, we have $f_{(i+1),P} = \alpha_{(i+1),P} + \theta y_P y_Q \beta_{(i+1),P}$ where

$$\alpha_{(i+1)} = ((x_P^3 + Ax_P + B)\alpha_{i,P} c_2 - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P})$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} + \beta_{i,P} c_2)/(x_Q - x_{(i+1),P}).$$

$Q)$ with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda' x + \nu'$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

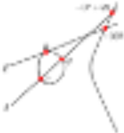


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\text{prod}} = \ell\ell'$ is $(\ell_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\text{quot}} = \ell/\ell'$ is $(\ell_{\text{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quot} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y-\lambda x-\nu}{y-\lambda' x-\nu'}$ gives $\frac{Y-\lambda X-\nu Z}{Y-\lambda' X-\nu' Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y-\lambda x-\nu)(y-\lambda' x-\nu')$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{(x^3+ax+b)+(\lambda x+\nu)(\lambda' x+\nu')}{(\lambda+\lambda')x+\nu+\nu'}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on $\mathbb{A}^2$) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
 1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
 2: for i from 0 to l − 1 do
 3:     if ($m_i = 1$) then
 4:         $A_R[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j + 1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_R[0]$,  $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le \mathrm{wt}(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,  $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda' x + \nu'$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

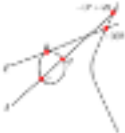


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\text{prod}} = \ell\ell'$ is $(\ell_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\text{quo}} = \ell/\ell'$ is $(\ell_{\text{quo}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$: projectifying $\ell/\ell' = \frac{y - \lambda x - \nu}{y - \lambda' x - \nu'}$ gives $\frac{Y - \lambda X - \nu Z}{Y - \lambda' X - \nu' Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda x - \nu)(y - \lambda' x - \nu')$, substituted the y^2 for $x^3 + ax + b$ and wrote $y =$ [illegible]. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These large costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$
 1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
 2: for $i$ from 0 to $l-1$ do
 3:     if ($m_i = 1$) then
 4:         $A_R[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j+1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_R[0]$,  $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \leq hw(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,  $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_P y_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P\gamma_i$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.
Multiplying all these together gives $f_{2i,P} = y_P\alpha_{2i,P} + \theta y_Q\beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2\gamma_i - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_Q^3 + Ax_Q + B)\theta\gamma_i\beta_{i,P}^2 + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{y_{iP} - 1}{x_{iP} - x_P}(x - x_P) + 1\right),$$

and so:

$$l(Q) = \theta y_Q - y_P\left(\frac{y_{i,P} - 1}{x_{iP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P\delta_i$, we have $f_{(i+1),P} = \alpha_{(i+1),P} + \theta y_P y_Q\beta_{(i+1),P}$ where

$$\alpha_{(i+1)} = ((x_Q^3 + Ax_Q + B)\alpha_{i,P}\delta_i - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P}),$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} - \beta_{i,P}\delta_i)/(x_Q - x_{(i+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda' x + \nu'$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

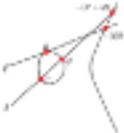


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\mathrm{prod}} = \ell\ell'$ is $(\ell_{\mathrm{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\mathrm{quo}} = \ell/\ell'$ is $(\ell_{\mathrm{quo}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quo} does not intersect E at $\mathcal{O}$; projectivising $\ell/\ell' = \frac{Y - \lambda X - \nu Z}{Y - \lambda' X - \nu' Z}$ gives $\frac{1 - \lambda X/Y - \nu Z/Y}{1 - \lambda' X/Y - \nu' Z/Y}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda x - \nu)(y - \lambda' x - \nu')$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{\ell\ell' - x^3 - ax - b - (\lambda x + \nu)(\lambda' x + \nu')}{-(\lambda + \lambda')x - \nu - \nu'}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than for the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G_2'$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,\psi(Q')}(P)$ representing a class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$
1: $R' \leftarrow Q'$,  $f \leftarrow 1$,  $j \leftarrow 0$
2: for $i$ from 0 to $l - 1$ do
3:     if ($m_i = 1$) then
4:         $A_R[j] \leftarrow R'$,  $A_f[j] \leftarrow f$,  $j \leftarrow j + 1$
5:     end if
6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,  $R' \leftarrow [2]R'$
7: end for
8: $R' \leftarrow A_R[0]$,  $f \leftarrow A_f[0]$
9: for ($j \leftarrow 1$; $j \le w(m) - 1$; $j{+}{+}$) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_R[j])}(P)$,  $R' \leftarrow R' + A_R[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$\ell(Q) = \theta y_Q - y_{k,P}\left(\frac{3x_{k,P}^2 + A}{2(x_{k,P}^3 + Ax_{k,P} + B)}(x_Q - x_{k,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2k,P}$. Multiplying all these together gives $f_{2k,P} = y_P\alpha_{2k,P} + \theta y_Q\beta_{2k,P}$ where

$$\alpha_{2k,P} = [\alpha_{k,P}^2 c_1 - (x_Q^3 + Ax_Q + B)\beta_{k,P}^2]/(x_Q - x_{2k,P})$$

and

$$\beta_{2k,P} = ((x_P^3 + Ax_P + B)\theta_k\beta_{k,P} + \alpha_{k,P}^2)/(x_Q - x_{2k,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{u_{k} - 1}{x_{k} - x_P}(x - x_P) + 1\right).$$

and so

$$\ell(Q) = \theta y_Q - y_P\left(\frac{u_{k,P} - 1}{x_{k} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_2$, we have $f_{(k+1),P} = \alpha_{(k+1),P} + \theta y_P y_Q\beta_{(k+1),P}$ where

$$\alpha_{(k+1)} = ((x_P^3 + Ax_P + B)\alpha_{k,P}c_2 - (x_Q^3 + Ax_Q + B)\beta_{k,P})/(x_Q - x_{(k+1),P}),$$

and

$$\beta_{(k+1),P} = (\alpha_{k,P} + \beta_{k,P}c_2)/(x_Q - x_{(k+1),P}).$$

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

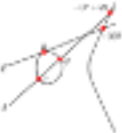


Figure 3.5: The functions ℓ and ℓ' on E.

the function $\ell_{\mathrm{num}} = \ell\ell'$ is $(\ell_{\mathrm{num}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P+Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\mathrm{den}} = \ell/\ell'$ is $(\ell_{\mathrm{den}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{den} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{Y-\lambda_1 X-\nu_1 Z}{Y-\lambda_2 X-\nu_2 Z}$ gives $\frac{Y-\lambda_1 X}{Y-\lambda_2 X}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y-\lambda_1 x-\nu_1)(y-\lambda_2 x-\nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y =$ [illegible]. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

than in the conventional left-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general $\mathbb{F}_{q^k}$-multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed addition steps for even k and ate-like pairings.

```
Input: $Q' \in G'_2$, $P \in G_1$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$, $m_{l-1} = 1$
Output: $f_{m,Q'}(P)$ representing a class in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$
 1: $R' \leftarrow Q'$,   $f \leftarrow 1$,   $j \leftarrow 0$
 2: for $i$ from 0 to $l - 1$ do
 3:     if ($m_i = 1$) then
 4:         $A_{R'}[j] \leftarrow R'$,   $A_f[j] \leftarrow f$,   $j \leftarrow j + 1$
 5:     end if
 6:     $f \leftarrow f^2 \cdot l_{\psi(R'),\psi(R')}(P)$,   $R' \leftarrow [2]R'$
 7: end for
 8: $R' \leftarrow A_{R'}[0]$,   $f \leftarrow A_f[0]$
 9: for ($j \leftarrow 1$; $j \le hw(m) - 1$; $j$++) do
10:     $f \leftarrow f \cdot A_f[j] \cdot l_{\psi(R'),\psi(A_{R'}[j])}(P)$,   $R' \leftarrow R' + A_{R'}[j]$
11: end for
12: return $f$
```

Parallelizing a single pairing

However, the right-to-left algorithm can be parallelized, and this could lead to more efficient implementations by taking advantage of many-core machines. Grabher, Großschädl, and Page [34, Algorithm 2] use a version of Algorithm 3

$$l(Q) = \theta y_Q - y_{k,P}\left(\frac{3x_{k,P}^2 + A}{2(x_{k,P}^3 + Ax_{k,P} + B)}(x_Q - x_{k,P}) + 1\right).$$

We write this as $\theta y_Q + y_P c_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2k,P}$. Multiplying all these together gives $f_{2k,P} = y_P \alpha_{2k,P} + \theta y_Q \beta_{2k,P}$ where

$$\alpha_{2k,P} = [\alpha_{k,P}^2 c_1 - (x_Q^3 + Ax_Q + B)\beta_{k,P}^2]/(x_Q - x_{2k,P})$$

and

$$\beta_{2k,P} = ((x_P^3 + Ax_P + B)\theta_k \beta_{k,P}^2 + \alpha_{k,P}^2)/(x_Q - x_{2k,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{y_{kP} - 1}{x_{kP} - x_P}(x - x_P) + 1\right).$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{y_{k,P} - 1}{x_{k,P} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P c_2$, we have $f_{(k+1),P} = \alpha_{(k+1),P} + \theta y_P y_Q \beta_{(k+1),P}$ where

$$\alpha_{(k+1)} = ((x_P^3 + Ax_P + B)\alpha_{k,P} c_2 - (x_Q^3 + Ax_Q + B)\beta_{k,P})/(x_Q - x_{(k+1),P})$$

and

$$\beta_{(k+1),P} = (\alpha_{k,P} - \beta_{k,P} c_2)/(x_Q - x_{(k+1),P}).$$

core idea

For $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$, don't use curve arithmetic but pairing $e(P, Q)$ to get overlap in orders!

Better suited for papers than slides

Computing pairings fast is quite technical.

general notice

Instead I describe the general approach, and leave all details out

general approach

1

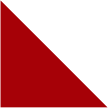
implement all tricks

that apply

2

benchmark speed

and finetune



3

fast pairings

0

take some literature

$$l(Q) = \theta y_Q - y_P y_{iP}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P r_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P\alpha_{2i,P} + \theta y_Q\beta_{2i,P}$ where

$$\alpha_{2i,P} = [\alpha_{i,P}^2 r_1 - (x_Q^3 + Ax_Q + B)\beta_{i,P}^2]/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_P^3 + Ax_P + B)\beta_{i,P}^2 + \alpha_{i,P}^2)/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line l is

$$y - y_P\left(\frac{u_{iP} - 1}{x_{iP} - x_P}(x - x_P) + 1\right),$$

and so

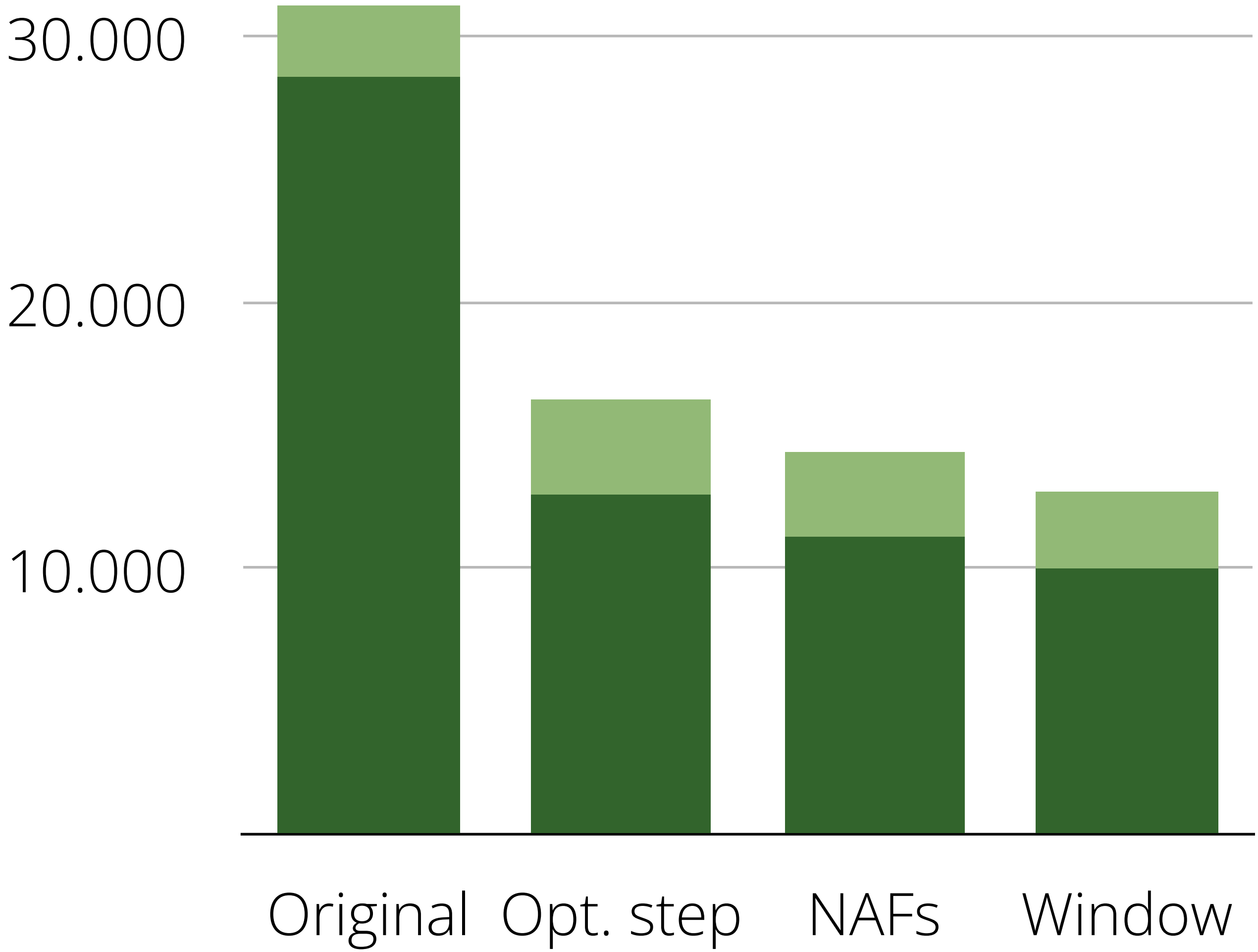
$$l(Q) = \theta y_Q - y_P\left(\frac{u_{i,P} - 1}{x_{iP} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P r_2$, we have $f_{(i+1),P} = \alpha_{i+1,P} + \theta y_P y_Q\beta_{i+1,P}$ where

$$\alpha_{(i+1)} = ((x_P^3 + Ax_P + B)\alpha_{i,P} r_2 - (x_Q^3 + Ax_Q + B)\beta_{i,P})/(x_Q - x_{(i+1),P})$$

and

$$\beta_{(i+1),P} = (\alpha_{i,P} - \beta_{i,P} r_2)/(x_Q - x_{(i+1),P}).$$


30.000
20.000
10.000
Original
Opt. step
NAFs
Window

3

fast pairings