A Variant of Miller's Formula and Algorithm

John Boxall[1], Nadia El Mrabet [2],
Fabien Laguillaumie[1] and Duc-Phong Le[3]

[1] LMNO - GREYC - University of Caen, France
[2] LIASD - University Paris 8, France
[3] Temasek Laboratories, National University of Singapore.

Yamanaka Hot Spring, December 15, 2010

Outline

Pairings

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ be three groups with the same order r. A pairing is a *non degenerate* and *bilinear map* :

$$e : G_1 \times G_2 \rightarrow G_3$$

Property

$$\forall j \in \mathbb{N}, e([j]P, Q) = e(P, Q)^j = e(P, [j]Q)$$

Computation of pairings

In cryptography, the sub groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of an elliptic curve, and $\mathbb{G}_3$ is a subgroup of a finite field. The more often used method to compute a pairing is the Miller's algorithm.

Several improvements for pairing based cryptography

Since the introduction of Miller's algorithm in cryptography, several optimizations have been made :

- tower fields extension,
- use of twisted curves,
- $\eta-$pairing, Ate pairing,
- new systems of coordinates,
- optimal pairings...

Our improvement of Miller's algorithm

The method

We work in a more general improvement of Miller's algorithm.
We propose a variant of Miller's algorithm which is generically faster than the usual version.

The classical Miller's algorithm is based on the equality :

$$f_{s+t,P} = f_{s,P} f_{t,P} \frac{\ell_{sP,tP}}{v_{(s+t)P}}.$$

where $f_{n,P}$ is the function with divisor $n[P] - [nP] - (n-1)[P_\infty]$.
We propose to work using another equality.

Our improvement of Miller's algorithm

The lemma

Lemma

For s and t two integers, up to a multiplicative constant, we have

$$f_{s+t,P} = \frac{1}{f_{-s,P} f_{-t,P} \ell_{-sP,-tP}}.$$

Our improvement of Miller's algorithm

The lemma

Lemma

For s and t two integers, up to a multiplicative constant, we have

$$f_{s+t,P} = \frac{1}{f_{-s,P} f_{-t,P} \ell_{-sP,-tP}}.$$

Proof

This lemma is proved by considering divisors.

$$\begin{aligned}
\mathrm{div}(f_{-s,P} f_{-t,P} \ell_{-sP,-tP}) &= (-s)[P] - [(-s)P] + (s+1)[P_\infty] \\
&\quad +(-t)[P] - [(-t)P] + (t+1)[P_\infty] \\
&\quad +[-sP] + [-tP] + [(s+t)P] - 3[P_\infty] \\
&= -(s+t)[P] + [(s+t)P] + (s+t-1)[P_\infty] \\
&= -\mathrm{div}(f_{s+t,P}),
\end{aligned}$$

Our improvement of Miller's algorithm

The notation

Before giving our version of Miller's algorithm, we introduce some notations :

- we use the lemma for $t = s$ or $t \in \{\pm 1\}$,
- we separate the computation of numerator and denominator in the equation : $f_{s+t,P} = \frac{1}{f_{-s,P} f_{-t,P} \ell_{-sP,-tP}}$ in $N\ell$ and $D\ell$.
- we use $\ell'_{-T,-P} = f_{-1,P} \ell_{-T,-P}$,

Our improvement of Miller's algorithm

The notation

The function $\ell'_{-T,-P} = f_{-1,P}\ell_{-T,-P}$

Using the new formulae, we have to compute $f_{-1,P}\ell_{-T,-P}$.
Even if $f_{-1,P}$ can be precomputed, it is more efficient to compute $\ell'_{-T,-P} = f_{-1,P}\ell_{-T,-P}$ instead of computing $f_{-1,P}$ and $\ell_{-T,-P}$ and taking the product.

Our improvement of Miller's algorithm

The notation

The function $\ell'_{-T,-P} = f_{-1,P}\ell_{-T,-P}$

Using the new formulae, we have to compute $f_{-1,P}\ell_{-T,-P}$.
Even if $f_{-1,P}$ can be precomputed, it is more efficient to compute $\ell'_{-T,-P} = f_{-1,P}\ell_{-T,-P}$ instead of computing $f_{-1,P}$ and $\ell_{-T,-P}$ and taking the product.

Exemple in affine coordinates

$$\ell'_{-T,-P} = \frac{y_Q + y_P}{x_Q - x_P} + \lambda.$$

Miller's algorithm and our improvement

The algorithm

Data: $s = \sum_{i=0}^{l-1} s_i 2^i$, $h = Hw(s)$, $Q \in E(\mathbb{F}')$ not a multiple of P
Result: $f_{s,P}(Q)$;
$f \leftarrow 1$, $T \leftarrow P$;
if $l + h$ *is odd* **then**
  $\delta \leftarrow 1$, $g \leftarrow f_{-1,P}$
end
else
  $\delta \leftarrow 0$, $g \leftarrow 1$;
end

Miller's algorithm and our improvement

The algorithm

for $i = l - 2$ **to** 0 **do**

1 **if** $\delta = 0$ **then**

$f \leftarrow f^2 (N\ell)_{T,T}$, $g \leftarrow g^2 (D\ell)_{T,T}$, $T \leftarrow 2T$, $\delta \leftarrow 1$;

2 **if** $s_i = 1$ **then**

$g \leftarrow g(N\ell')_{-T,-P}$, $f \leftarrow f(D\ell')_{-T,-P}$, $T \leftarrow T + P$, $\delta \leftarrow 0$;

end

end

3 **else**

$g \leftarrow g^2 (N\ell)_{-T,-T}$, $f \leftarrow f^2 (D\ell)_{-T,-T}$, $T \leftarrow 2T$, $\delta \leftarrow 0$;

4 **if** $s_i = 1$ **then**

$f \leftarrow f(N\ell)_{T,P}$, $g \leftarrow g(D\ell)_{T,P}$, $T \leftarrow T + P$, $\delta \leftarrow 1$;

end

end

end

return f/g

Analysis of our algorithm

The generic analysis

We compare the number of operations needed to compute $f_{s,P}(Q)$ using the classical Miller's algorithm and our.

In order to fix ideas, we use Jacobian coordinates associated to a short Weierstrass model $y^2 = x^3 + ax + b$, a, $b \in \mathbb{F}$.

We suppose that the Jacobian coordinates of P lie in $\mathbb{F}$ and that those of Q lie in some extension $\mathbb{F}'$ of $\mathbb{F}$ of whose degree is denoted by k.

We denote by m_a the multiplication by the curve coefficient a and respectively by m (M_k) and s (S_k) multiplications and squares in $\mathbb{F}$ ($\mathbb{F}'$).

Our improvement of Miller's algorithm

The generic analysis

Operation	Classical Miller	Modified Miller loop 1	Modified Miller loop 2
Doubling	$m_a + 8s$ $+(5+5k)m$ $+2S_k + 2M_k$	$m_a + 7s$ $+(5+3k)m$ $+2S_k + M_k$	$m_a + 7s$ $+(5+3k)m$ $+2S_k + M_k$
Addition	$4s + (8+5k)m$ $+2M_k$	$3s + (8+2k)m$ $+M_k$	$3s + (8+3k)m$ $+M_k$

FIGURE: Analysis of the cost of generic algorithm

Our improvement of Miller's algorithm

Curves with even embedding degree

A classical optimisation in pairing based cryptography is to consider elliptic curve with even embedding degree.

- Such curve admit a twist and it is possible to eliminate the computation of denominators.
- Another advantage is the use of tower extension of fields in order to improve the computation.

Our algorithm can be modified for such curves.

Our improvement of Miller's algorithm

Curves with even embedding degree

We replace the denominators $\ell_{-T,-T}$ and $\ell_{-T,-P}$ (updated in the function g) by their conjugates $\overline{\ell_{-T,-T}}$ and $\overline{\ell_{-T,-P}}$.
This operation transforms inversions into multiplications. The advantage is that we do not have to update the function g for our version of Miller's algorithm.

Our improvement of Miller's algorithm

Curves with even embedding degree

We replace the denominators $\ell_{-T,-T}$ and $\ell_{-T,-P}$ (updated in the function g) by their conjugates $\overline{\ell_{-T,-T}}$ and $\overline{\ell_{-T,-P}}$.
This operation transforms inversions into multiplications. The advantage is that we do not have to update the function g for our version of Miller's algorithm.
For exemple, in Jacobian coordinates, one has

$$\overline{(N\ell')_{-T,-P}} = \overline{\alpha_{Q,P}}(D\lambda)_{T,P} + (N\lambda)_{T,P}, \qquad \text{and}$$

$$\overline{(N\ell)_{-T,-T}} = 2Y_T(-y_Q Z_T^3 + Y_T) + (N\mu)_T(x_Q Z_T^2 - X_T).$$

Our improvement of Miller's algorithm

Curves with even embedding degree

Data: $s = \sum_{i=0}^{l-1} s_i 2^i$, $h = Hw(s)$, $Q \in E[r]$.
Result: An element f of $\mathbb{F}_{q^k}$ satisfying $f^{q^{k/2}-1} = f_{s,P}(Q)^{q^{k/2}-1}$
$f \leftarrow 1$, $T \leftarrow P$, ;
if $l + h$ *is odd* **then**
  $\delta \leftarrow 1$;
end
else
  $\delta \leftarrow 0$;
end

Our improvement of Miller's algorithm

Curves with even embedding degree

for $i = l - 2$ **to** 0 **do**

1 **if** $\delta = 0$ **then**

$f \leftarrow f^2 (N\ell)_{T,T}$, $T \leftarrow 2T$, $\delta \leftarrow 1$;

2 **if** $s_i = 1$ **then**

$f \leftarrow f\overline{(N\ell')_{-T,-P}}$, $T \leftarrow T + P$, $\delta \leftarrow 0$;

end

end

3 **else**

$f \leftarrow f^2\overline{(N\ell)_{-T,-T}}$, $T \leftarrow 2T$, $\delta \leftarrow 0$;

4 **if** $s_i = 1$ **then**

$f \leftarrow f(N\ell)_{T,P}$, $T \leftarrow T + P$, $\delta \leftarrow 1$;

end

end

end

return f

Our improvement of Miller's algorithm

Curves with even embedding degree

Quantity	Modified Miller (loop **1**)	Modified Miller (loop **3**)
Doubling	$m_a + 7s$ $+(5+k)m$ $+S_k + M_k$	$m_a + 7s$ $+(5+k)m$ $+S_k + M_k$
Addition	$3s + (8+k)m$ $+M_k$	$3s + (8+k)m$ $+M_k$

Our improvement of Miller's algorithm

Experiments

We ran some experiments comparing usual Miller with our variant when $k = 17$, $k = 18$ and $k = 19$.
In each case, the group order r has 192 bits and the rho-value $\rho = \frac{\log q}{\log r}$ is a little under 1.95, q being the cardinality of the base field.
Our curves were constructed using the Cocks-Pinch method.
For the computations, we used the NTL library and implemented the algorithms without any optimization on an Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16Ghz using Ubuntu Operating System 9.04.

Our improvement of Miller's algorithm

Experiments

k	Usual Miller	Our variant	Our variant with k even	Miller without
17	$0.0664s$	$0.0499s$	-	-
18	$0.0709s$	-	$0.0392s$	$0.0393s$
19	$0.0769s$	$0.0683s$	-	-

FIGURE: Timings

Conclusion

Our new version of Miller's algorithm works perfectly well for arbitrary embedding degree.

Potential applications :

- prime embedding degrees or, more generally, embedding degrees not of the form $2^i 3^j$.
- optimal pairings (Vercauteren, Hess)

Further work is needed to clarify this.

Thank you very much

for your attention.

Do you have any question ?