Delaying Mismatched Field Multiplications in Pairing Computations

Craig Costello

craig.costello@qut.edu.au
Queensland University of Technology

WAIFI 2010 - Istanbul, Turkey

Joint work with Colin Boyd, Juanma Gonzalez-Nieto, Kenneth Koon-Ho Wong

Pairing computation speeds: then and now

- **Then**:
  - 1993 Menezes' elliptic curve book : few minutes

...BIG GAP...

- **Now**:
  - 2009 Hankerson, Menezes, Scott: 4.01ms
  - April 2010 Naehrig, Niederhagen, Schwabe: 1.80ms
  - June 2010 Beuchat *et al.*: 0.94ms

So what happened in the big gap?

- Heaps of exciting protocol stuff has happened...
  ID-based encryption (IBE), ID-based key agreement, short signatures, group signatures, ring signatures, certificateless encryption, hierarchical encryption, predicate-based encryption, attribute-based encryption, ... and many more!!!

- Heaps of cool pairing optimizations has since 'followed'...
  - *Tate pairing instead of Weil pairing*
  - *denominator elimination*
  - *group choices and twisted curves*
  - *endomorphism rings and loop shortening*
  - *low rho-valued curves*
  - *pairing and towering-friendly fields*
  - *... and many more!!!*

- Many of the high-level optimizations on **elliptic curves (genus 1)** have been thoroughly explored
- Meanwhile, more neat ideas and notable optimizations continue to solidly improve the situation (Granger & Scott PKC'10, Benger & Scott WAIFI'10, ALNR with Edwards, etc)
- The time is ripe for 'lower-level' and implementation specific improvements
- Even though they're faster than a milli-second, some cryptographers still think they're slow in practice... so we will keep optimizing...

This work

- Targets one step in Miller's algorithm that hasn't received a great deal of attention
- Step where different degree extension fields are combined $\mathbb{F}_p, \mathbb{F}_{p^{k/d}}, \mathbb{F}_{p^k} \longrightarrow \mathbb{F}_{p^k}$.
- 'Replaces' higher degree extension field arithmetic with arithmetic in smaller subfields
- Ultimate goal: optimize the number of equivalent base field $\mathbb{F}_p$-operations

The embedding degree k

Must form a degree k field extension of $\mathbb{F}_q$ to find two order r subgroups and balance ECDLP and DLP

$$\mathbb{G}_1 = E[r] \cap \ker(\pi_q - [1]) = E(\mathbb{F}_q)[r], \qquad \text{(the base field)}$$

$$\mathbb{G}_2 = E[r] \cap \ker(\pi_q - [q]) \subset E(\mathbb{F}_{q^k})[r], \qquad \text{(the full extension field)}$$

The elements of $\mathbb{G}_2$ are much bigger than the elements of $\mathbb{G}_1$ (e.g. $k = 12$)

$$\mathbb{F}_{q^{12}} = \mathbb{F}_{q^4}(\alpha) = \mathbb{F}_{q^2}(\gamma) = \mathbb{F}_q(\beta)$$

$P \in \mathbb{G}_1$: [341746248540, 710032105147]

$Q \in \mathbb{G}_2$:

$[((502478767360 \cdot \beta + 1034075074191) \cdot \gamma + 342970860051 \cdot \beta + 225764301423) \cdot \alpha^2 + ((205398279920 \cdot \beta + 182600014119) \cdot \gamma + 860891557473 \cdot \beta + 435210764901) \cdot \alpha + (1043922075477 \cdot \beta + 566889113793) \cdot \gamma + 150949917087 \cdot \beta + 21392569319,$

$((654337640030 \cdot \beta + 744622505639) \cdot \gamma + 1092264803801 \cdot \beta + 895826335783) \cdot \alpha^2 + ((529466169391 \cdot \beta + 550511036767) \cdot \gamma + 985244799144 \cdot \beta + 554170865706) \cdot \alpha + (194564971321 \cdot \beta + 969736450831) \cdot \gamma + (579122687888 \cdot \beta + 581111086076)]$

The twisted curve

- Original curve is $E(\mathbb{F}_q) : y^2 = x^3 + ax + b$
- Twisted curve is $E'(\mathbb{F}_{q^{k/d}})$: $y^2 = x^3 + a\omega^4 x + b\omega^6$, $\omega \in \mathbb{F}_{q^k}$
- Possible degrees of twists are $d \in \{2, 3, 4, 6\}$
- $d > 2$ requires $a = 0$ or $b = 0$
- Twist $\Psi : E' \to E : (x', y') \to (x'/\omega^2, y'/\omega^3)$ induces $\mathbb{G}_2' = E'(\mathbb{F}_{q^{k/d}})[r]$ so that $\Psi : \mathbb{G}_2' \to \mathbb{G}_2$
- Instead of working with $Q \in \mathbb{G}_2$, a lot of work can be done with $Q' \in \mathbb{G}_2'$ defined over subfield $\mathbb{F}_{q^e} = \mathbb{F}_{q^{k/d}}$

$P \in \mathbb{G}_1$: $(341746248540, 710032105147)$

$Q' \in \mathbb{G}_2' = \Psi^{-1}(\mathbb{G}_2)$:

$((917087150949\beta + 25693192139) \cdot \omega^2, (878885791226\beta + 860765811110) \cdot \omega^3)$

Lite vs. full pairings

Miller-lite (Tate, twisted ate, η, etc)

$$e_r : \mathbb{G}_1 \times \mathbb{G}_2 \to \mu_r, \ (P, Q) \mapsto f_{r,P}(Q)^{\frac{q^k-1}{r}}.$$

Miller-full (ate, R-ate, ate$_i$, etc)

$$a_T : \mathbb{G}_2 \times \mathbb{G}_1 \to \mu_r, \ (Q, P) \mapsto f_{T,Q}(P)^{\frac{q^k-1}{r}}.$$

- Pairings require the computation of Miller functions $f_{m,R}(S)$
- Function $f_{m,R}$ is of degree m
- Constructions require $\lfloor \log_2 m \rfloor$ iterations of Miller's algorithm
- Most of the work is done in the first argument
- Tate needs $\lfloor \log_2 r \rfloor$ iters, ate needs $\lfloor \log_2 T \rfloor$ iters, $T \ll r$
- Trade-off is that more work in ate is done in larger field ($\mathbb{G}_2'$)

Miller-lite pairings

- The results in this paper are advantageous for Miller-lite pairings (bigger gap between P's coordinates and $\mathbb{F}_{q^k}$)
- Thus, from here on assume first arg. $P = (x_P, y_P) \in E(\mathbb{F}_q)$ (base field) and second arg. $Q = (x_Q, y_Q) \in E(\mathbb{F}_{q^k})$ (extension field)
- The pairing is computed as $e(P, Q) = f_{r,P}(Q)^{(q^k-1)/r}$, where $f_{r,P}(Q)$ would expand explicitly as

$$f_{r,P}(Q) = \sum_{i=0}^{r} \sum_{j=0}^{i} c_{i,j} \cdot x_Q^{i-j} y_Q^j,$$

  where the $c_{i,j}$'s are entirely P dependent, $c_{i,j} \in \mathbb{F}_q$.
- Indeterminate $f_{r,P}(x)$ has degree r (at least 160 bits), so must compute by building function and evaluating as we go...

Miller's algorithm for $e(P, Q) = f_{r,P}(Q)^{(q^k-1)/r}$

Input: P, Q and $r = (r_{\lfloor \log(r) \rfloor}, ..., r_0)_2$
Output: $f_{r,P}(Q)^{(q^k-1)/r}$

- $f \leftarrow 1$, $T \leftarrow P$
- **for** i **from** $\lfloor \log(r) \rfloor - 1$ **to** 0 **do**
  1. Compute $g = l$ in the chord-and-tangent doubling of T
  2. $T \leftarrow [2]T$
  3. $f \leftarrow f^2 \cdot g(Q)$
  4. **if** $r_i = 1$ **then**
     i. Compute $g = l$ in the chord-and-tangent addition of $T + P$
     ii. $T \leftarrow T + P$
     iii. $f \leftarrow f \cdot g(Q)$

     end if

end for: **return** $f \leftarrow f^{(q^k-1)/r}$

Miller's algorithm for $e(P, Q) = f_{r,P}(Q)^{(q^k-1)/r}$

State-of-the-art implementations employ low hamming-weight r values, so let's ignore additions (for now)

Input: P, Q and $r = (r_{\lfloor \log(r) \rfloor}, ..., r_0)_2$
Output: $f_{r,P}(Q)^{(q^k-1)/r}$

- $f \leftarrow 1$, $T \leftarrow P$
- **for** i **from** $\lfloor \log(r) \rfloor - 1$ **to** 0 **do**
  1. Compute $g = l$ in the chord-and-tangent doubling of T
  2. $T \leftarrow [2]T$
  3. $f \leftarrow f^2 \cdot g(Q)$
  4. **if** $r_i = 1$ **then**
     i. Compute $g = l$ in the chord-and-tangent addition of $T + P$
     ii. $T \leftarrow T + P$
     iii. $f \leftarrow f \cdot g(Q)$

     end if

end for: **return** $f \leftarrow f^{(q^k-1)/r}$

Miller's algorithm without the additions

Input: P, Q and $r = (r_{\lfloor \log(r) \rfloor}, \ldots, r_0)_2$
Output: $f_{r,P}(Q)^{(q^k-1)/r}$

- $f \leftarrow 1$, $T \leftarrow P$
- **for** i **from** $\lfloor \log(r) \rfloor - 1$ **to** 0 **do**
  1. Compute $g = l$ in the chord-and-tangent doubling of T
  2. $T \leftarrow [2]T$
  3. $f \leftarrow f^2 \cdot g(Q)$

end for: **return** $f \leftarrow f^{(q^k-1)/r}$

- Miller lite: Steps 1 and 2 are operations taking place in $\mathbb{F}_q$
- Step 3 takes place in $\mathbb{F}_{q^k}$
- $\mathbb{F}_{q^k}$ operations dominate computations, particularly as k gets larger
- let $\mathbf{m}_t, \mathbf{s}_t$ be cost of mul/squ in $\mathbb{F}_{q^t}$... if $t = 2^i 3^j$ then $\mathbf{m}_t = 3^i 5^j \mathbf{m}_1$ (Karatsuba, Toom-Cook multiplication)
- e.g. a multiplication in $\mathbb{F}_{q^{12}}$ costs $\mathbf{m}_{12} = 45\mathbf{m}_1$

A closer look at the Miller update $f^2 \cdot g(Q)$

Three steps in the update... just like traffic lights

i. $f \leftarrow f^2$

ii. Evaluate g at Q

iii. $f \leftarrow f \cdot g$

i. $f \in \mathbb{F}_{q^k}$;
$\rightarrow$ 1 full extension field multiplication (quadratic in $\mathbf{m}_1$)

ii. $g(x, y) = g_2 \cdot x + g_1 \cdot y + g_0$, $g_i \in \mathbb{F}_q$; multiplying g_i by coordinate of Q is computing $\mathbb{F}_q \cdot \mathbb{F}_{q^e}$;
$\rightarrow 2e$ multiplications in $\mathbb{F}_q$ (linear in $\mathbf{m}_1$)

iii. $g \in \mathbb{F}_{q^k}$ then looks something like
$g(x_Q, y_Q) = \hat{g}_2 \cdot \beta + \hat{g}_1 \cdot \alpha + g_0 \in \mathbb{F}_{q^k}$, with $g_1, g_2 \in \mathbb{F}_{q^e}$ and $g_0 \in \mathbb{F}_q$
$\rightarrow$ a bit awkward (g is usually sparse, f is not)... what to do???

What to do with f and g

- An example of f and g for a $d = 6$ sextic twist is used
  $$f = (f_{2,1} \cdot \alpha + f_{2,0}) \cdot \beta^2 + (f_{1,1} \cdot \alpha + f_{1,0}) \cdot \beta + (f_{0,1} \cdot \alpha + f_{0,0}) \in \mathbb{F}_{p^k},$$
  $$g = \hat{g}_2 \cdot \beta + \hat{g}_1 \cdot \alpha + g_0 \in \mathbb{F}_{q^k},$$
  where $f_{i,j}$'s and g_i's are in $\mathbb{F}_{q^{12}}$, α and β are algebraic (define extensions).
- Could just multiply adjust full extension field multiplication routine (and op count) accordingly
- Intuitively, we lose some of the "magic" of Karatsuba and Toom-Cook like techniques (difference between trivial coordinate-wise multiplication not so impressive)

Idea: What about not multiplying f by g in this iteration, and waiting for the next g' first before "touching" f

Perhaps $f \cdot (g \cdot g')$ *will beat* $(f \cdot g) \cdot g'$ *???*

What to do with f and g... cont

- Not actually as simple as $f \cdot (g \cdot g') \quad vs. \quad (f \cdot g) \cdot g'$ since g would have been absorbed into f and squared
- Should actually be $f \cdot (g^2 \cdot g') \quad vs. \quad (f \cdot g) \cdot g'$ which doesn't look as good!
- We've only touched $f \in \mathbb{F}_{q^k}$ once, but we have to do more to compute $g^2 \cdot g'$

Idea: Why don't we keep g as indeterminate... that way we don't even have to touch the $\mathbb{F}_{q^e}$ elements before $(g^2 \cdot g')$ is formed

All the work in forming the indeterminate $g^2 \cdot g'$ product will then be done is the base field $\mathbb{F}_q$

- Trade off: spending a lot more computations in $\mathbb{F}_q$ to avoid a computation in $\mathbb{F}_{q^k}$... potentially favorable, particularly for large k

Merging n iterations at a time

- If it is favorable to delay evaluation of g at Q and to delay the multiplication of f by $g(Q)$, why should we stop at delaying only once?
- The general case (merging n iterations at a time) looks like

for $i = \lfloor \log_{2^n}(r) \rfloor - 1$ **to** 0 **do**

Compute $g_{\text{prod}} = g_1^{2^{n-1}} g_2^{2^{n-2}} ... g_{n-1}^{2^1} g_n$

$T \leftarrow [2^n]T$ **(double n times)**

Evaluate g_{prod} at Q

$f \leftarrow f^{2^n} \cdot g_{\text{prod}}(Q)$

end for

- No more orange!!!

In case you missed any of that...

- Essentially, all we are doing is:
    - Loop unrolling Miller's algorithm (Granger, Page, Stam 2006) - supersingular characteristic 3 pairings
    - OR Miller's algorithm with window size n
    - OR loop parameter is written in 2^n-ary form, rather than binary form

This work vs. other work

- **for** $i = \lfloor \log_{2^n}(r) \rfloor - 1$ **to** 0 **do**

  Compute $g_{\mathrm{prod}} = g_1^{2^{n-1}} g_2^{2^{n-2}} \ldots g_{n-1}^{2^1} g_n$

  $T \leftarrow [2^n]T$ **(double** n **times)**

  Evaluate g_{prod} at Q

  $f \leftarrow f^{2^n} \cdot g_{\mathrm{prod}}(Q)$

  end for
- This work (1) vs. AfricaCrypt paper (2) - difference is the way g_{prod} is computed
  - (2) presents lengthy reduced explicit formulas
  - potentially cumbersome to implement
  - Herein we choose not to reduce $\rightarrow$ only slightly slower, but easier to implement

Non-reduced line products

- For $n = 2$: $g_{\mathrm{prod}} = (g_2 \cdot x + g_1 \cdot y + g_0)^2 \cdot (g_2' \cdot x + g_1' \cdot y + g_0')$
- Just expand g_{prod} in the trivial sense
- In the paper we generalize above product to an inderterminant product of n powers of lines: $g_{\mathrm{prod}}(x, y) = (g_2 \cdot x + g_1 \cdot y + g_0)^{2^n} \cdot (\hat{g}_2 \cdot x + \hat{g}_1 \cdot y + \hat{g}_0)^{2^{n-1}} \cdot \ldots \cdot (g_2' \cdot x + g_1' \cdot y + g_0')$
- Expand and reduce modulo $y^2 = x^3 + ax + b$ to give $g_{\mathrm{prod}} = h(x) + \hat{h}(x) \cdot y$
- Carefully keep track of optimal operation count to evaluate expanded version... assuming inputs of $(g_2, g_1, g_0) \in \mathbb{F}_p$ tuples
- Cost to get from g_{prod} to next $g'_{\mathrm{prod}} = g^2_{\mathrm{prod}} \cdot (g_2 \cdot x + g_1 \cdot y + g_0)$ and generalize...

The total cost of n merged iterations

$$\texttt{cost}_n = \left[6(2^n-1)+2\right]e\mathbf{m}_1 + \left[(n+1)(m+s\Omega)+3n(\Omega-6) + 3(2^n-1)((2^{n+1}-3)\Omega+12)\right]\mathbf{m}_1 + (1+(n+1)\Omega)\mathbf{m}_k,$$

- Plug in paramters (k, e, Ω) and minimize over n
- If $\texttt{cost}_{n>0}$ significantly better than $\texttt{cost}_0$ then speedup
- $k = 2^i \cdot 3^j \rightarrow \mathbf{m}_k = 3^i \cdot 5^j \mathbf{m}_1$ (all in terms of base field operations)

Operation counts and optimal n

k	D	m, s	$\mathbb{F}_{p^u} \subseteq \mathbb{F}_{p^e} \subset \mathbb{F}_{p^k}$	$\Omega = 1$ ($\mathbf{s} = \mathbf{m}$) $N = 0$ count	$\Omega = 1$ ($\mathbf{s} = \mathbf{m}$) **Optimal N** count	$\Omega = 0.8$ ($\mathbf{s} = 0.8\ \mathbf{m}$) $N = 0$	$\Omega = 0.8$ ($\mathbf{s} = 0.8\ \mathbf{m}$) **Optimal N** count
12	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^2} \subset \mathbb{F}_{p^{12}}$	103	**1** 96.5	92.6	**1** 85.5
14	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^7} \subset \mathbb{F}_{p^{14}}$	155	**1** 148	140.4	**1** 132.8
16	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^4} \subset \mathbb{F}_{p^{16}}$	180	**1** 159.5	162.2	**1** 141.1
18	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^3} \subset \mathbb{F}_{p^{18}}$	165	**1** 145.5	148.6	**1** 128.5
20	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^{10}} \subset \mathbb{F}_{p^{20}}$	254	**1** 217.5	229	**1** 191.9
22	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^{11}} \subset \mathbb{F}_{p^{22}}$	428	**1** 363	386.8	**1** 321.2
24	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^4} \subset \mathbb{F}_{p^{24}}$	287	**1** 239.5	258.6	**1** 210.5
26	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^{13}} \subset \mathbb{F}_{p^{26}}$	581	**1** 482.5	525	**1** 425.9
28	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^7} \subset \mathbb{F}_{p^{28}}$	420	**1** 347	378.8	**1** 305.2
30	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^{10}} \subset \mathbb{F}_{p^{30}}$	409	**1** 333.5	368.6	**1** 292.5
32	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^8} \subset \mathbb{F}_{p^{32}}$	512	**1** 418.5	461.8	**1** 367.7
34	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^{17}} \subset \mathbb{F}_{p^{34}}$	961	**2** 775.3	867.8	**2** 678.7
36	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^6} \subset \mathbb{F}_{p^{36}}$	471	**1** 382.5	424.6	**1** 335.5
38	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^{19}} \subset \mathbb{F}_{p^{38}}$	1187	**2** 936.7	1071.6	**2** 817.9
40	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^{10}} \subset \mathbb{F}_{p^{40}}$	732	**2** 585.6	660.2	**2** 510.5
42	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^7} \subset \mathbb{F}_{p^{42}}$	683	**2** 536.7	615.6	**2** 465.9
44	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^{11}} \subset \mathbb{F}_{p^{44}}$	1220	**2** 916.3	1099.6	**2** 792.5
46	1	2, 8	$\mathbb{F}_p \subset \mathbb{F}_{p^{23}} \subset \mathbb{F}_{p^{46}}$	1712	**2** 1308.3	1544.8	**2** 1137.7
48	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^8} \subset \mathbb{F}_{p^{48}}$	835	**2** 643.3	752.6	**2** 557.5
50	3	2, 7	$\mathbb{F}_p \subset \mathbb{F}_{p^{25}} \subset \mathbb{F}_{p^{50}}$	1073	**1** 881.5	970.2	**1** 778.1

Simple algorithm description

for $i = l_n - 2$ **to** 0 (loop parameter is base 2^n)

  Initialize $G(x, y) = h(x) + \hat{h}(x) \cdot y = 1$

  for $j = 1$ **to** n

    Compute $[(g_{n,2}, g_{n,1}, g_{n,0}), T] = \texttt{MillerDBL}(T)$

    Compute $G(x, y)^2 \cdot (g_{n,2} \cdot x + g_{n,1} \cdot y + g_{n,0})$

    $f \leftarrow f^2$

  end for

  Evaluate $G(x_Q, y_Q)$

  $f \leftarrow f \cdot G$

  if $m_i \neq 0$

    Compute $[g, T] = \texttt{MillerADD}(T, [m_i]R)$

    $f \leftarrow f \cdot f_{[m_i]R} \cdot g$

  end if

end for

return f

- $\texttt{MillerDBL}(T)$ and $\texttt{MillerADD}(T)$ same as always
- Often code is minor amendment to standard DBL and ADD routines

Summary

- An alternative to the explicit formulas in AfricaCrypt paper
- Only slight speed loss, **but** implementation is much easier
- Old code could be modified with injection of conceptually simple subroutine

Related work

- Minor add on to results: Miller-lite (Tate-like) pairings still in use for Type 1,2,4 pairings. Compression not always available so (x_Q, y_Q) are full extension field elements $\rightarrow$ future ePrint version
- Technique most powerful in context of fixed argument pairings, where P is a long-term secret key and precomputation is available in affine coordinate (http://eprint.iacr.org/2010/342 - "Fixed Argument Pairings")
- Actual implementations coming in future

Thanks