

Designing a code generator of Pairing Based Cryptography functions¹

Luis J. Dominguez Perez and Mike Scott

Dublin City University. School of Computing.

ldominguez.computing.dcu.ie.

C.S.I. Workshop on Coding & Cryptography.
Cork 2010. UCC.

May 17th-18th, 2010.



¹Or how to fit your thesis in 23 slides

Outline

- 1 Preliminaries
 - Tate-based pairings
- 2 Addition chains
- 3 $F.E.$ and G_2
 - Final Exponentiation
 - Scalar-point Multiplication
- 4 AIS
- 5 Code sequence
- 6 Conclusion

Motivation

- ▶ Pairing-Based Cryptography has become relevant in industry mainly because of the increasing interest in Identity-Based protocols, but it is more complex to implement than RSA, for example.
- ▶ We present a tool for automatically generating optimized code for pairing functions.
- ▶ Our cryptographic compiler chooses the most appropriate pairing function for the target family of curves, and constructs its code. It also generates optimized code for the final exponentiation and some supportive functions.

Notation

A **pairing** is a map: $G_1 \times G_2 \rightarrow G_T$.

Let $E(\mathbb{F}_p)$ be an elliptic curve and $r \mid \#E(\mathbb{F}_p)$. Let k be the embedding degree of E with respect to r .

Let e be a divisor of k and $\phi : E'(\mathbb{F}_{p^d}) \rightarrow E(\mathbb{F}_{p^k})$ where, $d = \frac{k}{e}$, $e \in \{2, 3, 4, 6\}$ and d -minimal, then E' is said to be a **twist** of E of degree e . And t is the trace of the Frobenius.

Let $\pi_p : \overline{\mathbb{F}}_p \rightarrow \overline{\mathbb{F}}_p$, $x \mapsto x^p$ is the p -power Frobenius map.

Let $\psi^i = \phi \pi^i \phi^{-1}$ be an homomorphism which maps points on the twist $E'(\mathbb{F}_{p^d})$ to the points of the curve $E(\mathbb{F}_{p^k})$.



The Tate pairing

Let $P \in E(\mathbb{F}_p)[r]$ and $Q \in E(\mathbb{F}_{p^k})$, a non degenerate bilinear Tate pairing can be defined as a map:

The Tate pairing

$$e_r : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) \rightarrow \mathbb{F}_{p^k}^*/(\mathbb{F}_{p^k}^*)^r$$

$$: (P, Q) \mapsto f_{r,P}(D)^{(p^k-1)/r}$$

This value of the pairing is in an equivalence class, $\mathbb{F}_{p^k}^*/(\mathbb{F}_{p^k}^*)^r$, to obtain a unique representative of the class (and to make it bilinear), we raise the value of the pairing to $(p^k - 1)/r$.

The ate Pairing

We denote $G_1 = E[r] \cap \text{Ker}(\pi_p - [1])$, $G_2 = E[r] \cap \text{Ker}(\pi_p - [p])$.

Let $T = t - 1$. For $Q \in G_2$ and $P \in G_1$ the ate pairing can be defined as:

The ate pairing

$$e_T : (Q, P) \longmapsto f_{T,Q}(P)^{(p^k-1)/r}$$

if $T^k \equiv 1 \pmod{r^2}$.

The R-ate Pairing

The R-ate pairing is a generalization of the ate and ate_i pairing, it takes three (two) short *Miller loops* to calculate the pairing.

From the ate_i pairing, $T_i \equiv p^i \pmod{r}$. Then, $T_A = a \cdot T_B + b$.

The definition of the *R-ate* pairing is as follows:

The R-ate pairing

$$e_{A,B}(P, Q) = f_{a,BP}(Q) \times f_{b,P}(Q) \times G_{aBP,bP}(Q)$$

Pairing comparison

This pairing is not always bilinear. We have to find a proper (A,B) .

Miller-length in iterations		
Tate	$e_r(P, Q)$	377
Ate	$e_t(P, Q)$	254
R-ate	$e_{A,B}(P, Q)$	61

Table: Comparison of miller-loop length for KSS: $k = 18$

In this case, we got a Miller length of $1/6$ compared to the Tate pairing.

for a $\text{Log}_2(p) \approx 512$

Intro to Addition Chains

- ▶ An **Addition Chain** for a given integer e is a sequence $U = \{u_0, u_1, u_2, \dots, u_l\}$ such that $u_0 = 1$, $u_l = e$ and $u_k = u_i + u_j$ for $k \leq l$ and some i, j with $0 \leq i \leq j$.
- ▶ Consider the following *Fibonacci* sequence: $\{1, 2, 3, 5, 8, 13, 21\}$. This is an addition chain for $e = 21$ which contains 7 elements.
- ▶ Now, consider $e = 34$: $\{1, 2, 3, 5, 8, 13, 21, 34\}$, but we have a better addition chain: $\{1, 2, 4, 8, 16, 32, 34\}$.

Addition Sequences

- ▶ Easy? Consider finding the *shortest* possible addition chain for $e = 2^{126} + 2^{123} + 2^{94} + 1$
The problem now becomes very tricky!... isn't it?
- ▶ Now, given a list of integers $\Gamma = \{v_1, \dots, v_l\}$ where $v_l \geq v_i$ for all $i = 1, \dots, l - 1$, an **Addition Sequence** for Γ is an addition chain for v_l containing all elements of Γ . The last element of the sequence is the exponent $e = v_l$.
- ▶ Let $\Gamma = \{1, 5, 12\}$, an addition sequence for Γ is the following:
 $\{\underline{1}, 2, 4, \underline{5}, 8, \underline{12}\}$.

Addition Sequences II

- ▶ Again, consider finding a short addition sequence for $\#\Gamma > 100$ and $\text{Log}_2(\Gamma_1) > 32 \dots$ not funny with pen-and-paper.

- ▶ Q: Where do we need such a large addition sequences?

Addition Sequences II

- ▶ Again, consider finding a short addition sequence for $\#\Gamma > 100$ and $\text{Log}_2(\Gamma_1) > 32 \dots$ not funny with pen-and-paper.
- ▶ Q: Where do we need such a large addition sequences?
- ▶ A: Multi-exponentiation techniques*.

Addition Sequences II

- ▶ Again, consider finding a short addition sequence for $\#\Gamma > 100$ and $\text{Log}_2(\Gamma_1) > 32 \dots$ not funny with pen-and-paper.
- ▶ Q: Where do we need such a large addition sequences?
- ▶ A: Multi-exponentiation techniques*.

Definition

- ▶ One of the most expensive operations in the Tate-based pairings is the final exponentiation by $(p^k - 1)/r$. This computation eliminates the r -th powers and returns a unique r -th roots of unity, also providing bilinearity to the function.
- ▶ Nowadays, we separate the final exponentiation in easy and hard parts:

$$(p^k - 1)/r \Rightarrow \underbrace{(p^{\frac{k}{2}} + 1) \cdot (p^d - 1)}_{\text{easy}} \cdot \underbrace{\frac{\Phi_k(p)}{r}}_{\text{hard}}.$$

We can use simple exponentiation and the p -power Frobenius map for the easy part.

The hard part

- ▶ Using a base- p representation of the hard part, we can set:

$$f^{\frac{\Phi_k(p)}{r}} = f^{\lambda_0} \cdot (f^{\lambda_1})^p \cdot (f^{\lambda_2})^{p^2} \dots (f^{\lambda_{n-1}})^{p^{n-1}}$$

- ▶ For the KSS curves with $k = 18$, the polynomial representation is:

$$\lambda_5 = 49/3x^2 + 245/3x + 343/3.$$

$$\lambda_4 = 7/3x^6 + 35/3x^5 + 49/3x^4 + 112/3x^3 + 581/3x^2 + 784/3x.$$

$$\lambda_3 = -5/3x^7 - 25/3x^6 - 35/3x^5 - 29x^4 - 150x^3 - 203x^2 + 18.$$

$$\lambda_2 = -49/3x^5 - 245/3x^4 - 343/3x^3 - 931/3x^2 - 4802/3x - 6517/3.$$

$$\lambda_1 = 14/3x^6 + 70/3x^5 + 98/3x^4 + 91x^3 + 469x^2 + 637x.$$

$$\lambda_0 = -x^7 - 5x^6 - 7x^5 - 62/3x^4 - 319/3x^3 - 434/3x^2 + 1.$$

Addition chain for the FE

- ▶ In practice, we let $3 \cdot \lambda$. Hence, $\Gamma = \{3, 5, 7, 14, 15, 21, 25, 35, 49, 54, 62, 70, 87, 98, 112, 245, 273, 319, 343, 434, 450, 581, 609, 784, 931, 1407, 1911, 4802, 6517\}$.
- ▶ Then, we set $x_0 = f \cdot \frac{1}{f^{x_7}}$, $x_1 = \pi^3\left(\frac{1}{f^{x_7}}\right)$, and so on.

Fast hashing to G_2

- ▶ To get a point of order r in G_2 (ate and the R-ate pairings): hash to a random point on E' , then perform multiplication by c , a large cofactor.
- ▶ This cofactor multiplication is time-expensive.
- ▶ In short, we do the same, plus a few extra transformations, and we get:

$$\begin{aligned}
 [3.c(x)]P &= (-5x^7 - 26x^6 - 98x^5 - 381x^4 - 867x^3 - 1911x^2 - 5145x - 5774).P \\
 &+ (-5x^7 - 18x^6 - 38x^4 - 323x^3 - 28x^2 + 784x).\psi(P) \\
 &+ (-5x^7 - 18x^6 - 38x^4 - 323x^3 + 1029x + 343).\psi^2(P) \\
 &+ (-11x^6 - 70x^5 - 98x^4 - 176x^3 - 1218x^2 - 2058x - 686).\psi^3(P) \\
 &+ (28x^2 + 245x + 343).\psi^4(P)
 \end{aligned}$$

Addition chain for G_2

- ▶ For this case, $\Gamma = \{5, 11, 18, 26, 28, 38, 70, 98, 176, 245, 323, 343, 381, 686, 784, 867, 1029, 1218, 1911, 2058, 5145, 5774\}$

- ▶ Then, we set $x_0 = [x^7](-Q) + \psi([x^7](-Q)) + \psi^2([x^7](-Q))$

Methods to solve an addition chain/sequence

- ▶ Finding the shortest addition chain for a given positive integer is an NP-complete problem.
- ▶ We have methods available to find a sort addition chain:
 - ▶ Bos and Coster propose a “Makesequences”.
 - ▶ Bernstein propose a binary method using xor’s to solve an addition sequence.
 - ▶ Cruz-Cortez et al. propose an AIS for addition chains.
 - ▶ We modified the previous for addition sequences.

Original method

In general:

- ▶ Let S be a matrix with G cloned $\Gamma \cup \{1, 2\}$ proto-sequences
- ▶ Repeat for M times
- ▶ $A \leftarrow$ average size of the S sequences
- ▶ If $\#S_j > A$, then:
 - ▶ Mutate in the lower part

Otherwise:

- ▶ Mutate in the higher part

Our modifications to the method

- ▶ Mutation:
 - ▶ Remove a random element not in Γ
 - ▶ If a random integer can be constructed with the sequence or by a doubling, then we add it, otherwise, the restore the removed element.

We compared the AIS and binary methods with a long sequence: the final exponentiation of the KSS curves with $k = 36$ has $\#\Gamma = 69$, $\lg_2(v_l) = 26$

- ▶ Binary method: 363 elements non-improve-able
- ▶ AIS method: 213 elements in only 10 generations.

Vector Addition Chain

A **Vector Addition Chain** is the shortest possible list of vectors where each vector is the addition of two previous vectors. The last vector contains the last exponent e .

Let V be a vector chain, let m be the dimension of every vector. The vector addition chain starts with $V_{i,j} = 1$ for $i = 0 \dots m - 1$: $[1, 0, 0, \dots, 0], [0, 1, 0, \dots, 0], \dots, [0, \dots, 0, 1]$ and continue until $V_{j,1} = e$ with j typically $> m$.

A vectorial chain for $\Gamma = \{12, 5, 1\}$ is $\{[1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [2, 0, 0], [4, 0, 0], [0, 2, 0], [2, 2, 0], [4, 4, 0], [8, 4, 0], [12, 4, 0], [12, 5, 1]\}$

Code construction

Olivos presented a method to convert the vector addition chain into a sequence of operations.

t_0	\leftarrow	$x_{27} \cdot x_{28}$
t_1	\leftarrow	$t_0 \cdot x_{24}$
t_2	\leftarrow	$t_0 \cdot t_0$
t_3	\leftarrow	$t_2 \cdot x_{28}$
t_4	\leftarrow	$t_3 \cdot x_{26}$
t_5	\leftarrow	$x_8 \cdot x_{26}$
t_6	\leftarrow	$t_4 \cdot t_4$
t_7	\leftarrow	$t_6 \cdot t_1$
t_8	\leftarrow	$x_{15} \cdot x_{25}$
t_9	\leftarrow	$x_{25} \cdot x_{25}$
...		

Table: Final Exponentiation Code from the Olivos and Scott et al. Method. KSS Curve: $k = 18$

t_0	\leftarrow	$x_{27} \cdot x_{28}$
t_1	\leftarrow	$t_0 \cdot x_{24}$
t_0	\leftarrow	$t_0 \cdot t_0$
t_0	\leftarrow	$t_0 \cdot x_{28}$
t_0	\leftarrow	$t_0 \cdot x_{26}$
t_5	\leftarrow	$x_8 \cdot x_{26}$
t_0	\leftarrow	$t_0 \cdot t_0$
t_3	\leftarrow	$t_0 \cdot t_1$
t_1	\leftarrow	$x_{15} \cdot x_{25}$
t_0	\leftarrow	$x_{25} \cdot x_{25}$
...		

Table: Final Exponentiation Code for a KSS Curve: $k = 18$ with a Reduced number of t_i Elements.

More on code construction

- ▶ We generate the code in a $3 \times m$ matrix to easily manipulate it and generate the appropriate code for the desired multiprecision library.
- ▶ For recycling the x_i 's variables and compute them, we can either have them *in-place* or by an assignment *just-in-time* before the operation.
- ▶ The code is competitive in code size and memory use against hand-crafted code.

▶ text

Contribution

- ▶ Our cryptographic compiler makes use of several meta-programming techniques and we have support for several multiprecision libraries. We have, however, only tested in Miracl and RELIC, and obviously in Magma, which is the language where it is constructed.
- ▶ Next step on this line of research is to design an automatic finite field arithmetic generator for different multiprecision libraries, as well, support for other pairing constructions.
- ▶ Speed improvements in the addition-chain generation are also welcome.

text