Automatic generation of optimised Cryptographic Pairing functions' code

Luis J. Dominguez P. and Michael Scott

Dublin City University. School of Computing.

ldominguez.computing.dcu.ie.

SPEED-CC, Berlin.

October 12^{th}-13^{th}, 2009.

CONACYT
Consejo Nacional de Ciencia y Tecnología

Claude Shannon Institute
Discrete Mathematics, Coding, Cryptography and Information Security
www.shannoninstitute.ie

DCU

Outline

DCU

Automatic generation of pairing functions' code

Motivation:

- Pairing based cryptography is more complex to understand and to implement than, for example, the RSA.
- If a higher security level is desired, a different pairing friendly curve should be selected. For optimisation, it will need a different implementation.

DCU

Definition

The computation of pairings basically involves two groups, G_1 and G_2. These groups are finite, cyclic, additively-written groups and at least one of which is of prime order r.

The pairing will take an element from each of the two groups and map them to a group, denoted G_T, which is a finite, cyclic, multiplicatively-written group also of prime order r.

DCU

Definition, II.

A useful cryptographic pairing satisfies the following properties:

- Bilinearity:
  For all $P, P' \in G_1$ and all $Q, Q' \in G_2$, one has: $e(P + P', Q) = e(P, Q) \times e(P', Q)$ and $e(P, Q + Q') = e(P, Q) \times e(P, Q')$
- Non-degeneracy:
  For all $P \in G_1$ with $P \neq 0$, there is some $Q \in G_2$ such that $e(P, Q) \neq 1$.
  For all $Q \in G_2$ with $Q \neq 0$, there is some $P \in G_1$ such that $e(P, Q) \neq 1$.
- Computability: can be easily evaluated.

DCU

Tate pairing

The Tate pairing was introduced as a general pairing on Abelian varieties over local fields. Lichtenbaum gave an application of this pairing to the Jacobians of curves over local fields.

Let $P \in E(\mathbb{F}_p)[r]$ and let $Q \in E(\mathbb{F}_{p^k})$. Let $f_{a,P}$ be a function with a divisor $(f_{a,P}) = a(P) - (aP) - (a-1)(\mathcal{O})$ for $a \in \mathbb{Z}$.

A non-degenerate, bilinear Tate pairing can be defined as a map:

Definition

$$e_r : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) \longrightarrow \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$$
$$(P, Q) \mapsto \langle P, Q \rangle_r = f_{r,P}(Q)$$

ate pairing

The ate pairing is a variant of the Tate pairing and it is a generalisation of the Eta pairing on ordinary pairing-friendly elliptic curves.

We denote $G_1 = E[r] \cap Ker(\pi_p - [1])$, $G_2 = E[r] \cap Ker(\pi_p - [p])$.
Let $T = t - 1$. Let $N = gcd(T^k - 1, p^k - 1)$, $T^k - 1 = LN$.

The ate pairing takes $Q \in G_2[r]$ and $P \in G_1$.

ate pairing, II.

Definition

The ate pairing is defined as:
$e_T : (Q, P) \longmapsto f_{T,Q}(P)^{c_T(p^k-1)/N}$

where $c_T = \sum_{i=0}^{k-1-i} p^i \equiv kp^{k-1} \bmod r$. The ate pairing is a bilinear non-degenerate pairing if $r \nmid L$.

In practice, the reduced ate pairing: $f_{T,Q}(P)^{(p^k-1)/r}$ is preferred.

R-ate pairing

The R-ate pairing introduced by Lee, Lee and Park is a generalisation of the ate and ate$_i$ pairing.

Let $T_i \equiv p^i \bmod r$ and $T_j \equiv p^j \bmod r$. We define $A = a.B + b$, where $T_i = a.T_j + b$.

The definition of the R-ate pairing with $A, B, a, b, \in \mathbb{Z}$ and non-trivial is as follows:

Definition

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

R-ate Pairing, II

Generally this definition does not always give a bilinear and non-degenerate pairing. For efficiency, we look for a combination of A and B that would give the shortest Miller loop.

The R-ate pairing algorithm uses a and b as follows:

- $m_1 = max\{a, b\}$,
- $m_2 = min\{a, b\}$,
- $f\{a, b\}, \{a, b\}Q = \{m_1, m_2\}$.

R-ate Pairing, III

The three (two) Miller loop calls are:

- $M(Q, P, m_2)$,
- $M(m_2 Q, P, c)$,
- $M(Q, P, d)$.

where the parameters for these Miller function calls are as follows:

- $m_1 \leftarrow max\{a, b\}$,
- $m_2 \leftarrow min\{a, b\}$,
- $c \leftarrow [\frac{m_1}{m_2}]$,
- $d \leftarrow m_1 - c \cdot m_2$.

DCU

R-ate pairing, IV.

KSS curves with embedding degree $k = 18$

Miller-length in iterations						
#	a	b	loops	Ham	m_2	c
1	3	x	23	4	0	23
2	$5/7x$	$1/7x^2$	47	6	24	23
3	$8/7x$	$3/7x^2$	48	7	25	23
...						
10	$3/7x$	$2/7x$	23	4	23	0
11	$2/7x$	$3/7x$	23	4	23	0
12	$3/14x$	$1/14x^2$	46	8	23	23
...						
15	$3/49x^2$	$5/49x^2$	47	5	47	0

Table: KSS: $k = 18$ Curves A,B parameters

The addition-chain

Definition: addition chain

An *addition chain* for a given number e is a sequence $U = (u_0, u_1, u_2, \ldots, u_\ell)$ such that $u_0 = 1, u_\ell = e$ and $u_k = u_i + u_j$ for some i, j with $0 \leq i \leq j < k \leq \ell$.

Finding a minimal addition chain for a given positive integer e is an NP-complete problem.

Definition: addition sequence

Given a list of integers $\Gamma = \{v_1, .., v_\ell\}$ where $v_\ell > v_i$ $\forall i = 1, .., \ell - 1$, an addition sequence for Γ is an addition chain for v_ℓ containing all elements of Γ.

Solving an addition-chain

Definition: multi-addition chain

Given a list of integers $\Lambda = \{s_1, .., s_\ell\}$ where $s_\ell > s_i$ $\forall i = 1, .., \ell - 1$, a multi addition chain for Λ is a set containing multiple addition chains in which some s_i are common.

A multi-addition-chain is a generalisation of the concept of the addition chains, which can be used to perform the final exponentiation.

To construct the multi-addition-chain we modify the Cruz-Cortéz et al. method, which is designed to generate a simple addition-chain for the RSA method using "Artificial Immune Systems".

The final exponentiation

One of the most expensive operations in the pairing computation is the final exponentiation by $(p^k - 1)/r$ in the extension field $\mathbb{F}_{p^k}$. This is required in the computation of the Tate, ate and R-ate pairings on ordinary elliptic curves.

Usually one separates the exponent into 3 pieces: $(p^k - 1)/r \Rightarrow (p^{\frac{k}{2}} + 1) \cdot (p^{\frac{k}{2}} - 1)/\Phi_k(p) \cdot (\Phi_k(p))/r$. The first 2 parts can be easily computed using the Frobenius exponentiation. The remaining, using the Scott et al.[1] method.

The final exponentiation, II

At the end of the Scott et al.[1] method, one uses the multi addition-chain method described in the last section and then generates the code using vector chains with the Olivos method.

Exponentiation in RSA using addition chains usually requires the shortest possible addition chain.

For the final exponentiation method, one may prefer a longer chain if it contains a greater number of doublings and lower number of additions, as it will generate code with more squarings and less multiplications.

Hashing to G_2[r].

Implementing identity-based protocols using ordinary pairing-friendly elliptic curves requires two groups, at least one of which is to be of order r.

If there is a requirement to hash to a point in G_2 of order r, then the operation becomes much more complex. Thanks to the use of a twisted curve, the operation cost is not exorbitant, but still need to be addressed.

DCU

Hashing to $G_2[r]$, II.

The Scott et al.[2] method for fast hashing to G_2 requires the use of our addition chains method. In this case, the code will generate a doubling or addition depending on the chain.

In the final exponentiation case, the operations are squarings and multiplications in $E(\mathbb{F}_{p^k})$. Here the operations are doublings or additions of a point in a large subgroup of points on a curve. A proper chain selection may lead to shorter computation times.

DCU

- Working demo... ?

Contribution and status

We have adapted several construction methods for automatically generate cryptographic pairing functions' code to ease the job of the protocol implementer.

Our tool can be extended to analyze different constraints such as low memory, high bandwidth cost, or higher speed requirements (i.e. more memory use).

A lot of work still need to be done to fully automate this tool. Families of pairing friendly curves with different discriminants can be added, finite field arithmetic on the fly, cross-platform compiling, among others.

DCU

Final thoughts

Open questions:

- How to generate an optimal addition chain,
- is addition faster than doubling in each scenario?

Future work:

- Parallel computation of the mini Miller loops at the R-ate pairing,
- try other method for the addition chain creation,
- define a method for the towering construction of the finite field extensions,
- adapt the arithmetic to the scenario.