Low-cost addition–subtraction sequences for the final exponentiation in pairings

Guzman-Trampé, Cruz-Cortés, Dominguez Perez, Ortiz-Arroyo, and Rodriguez-Henríquez

CIMAT CONACYT

CRYPTO-CO, Julio 9 de 2016

Exponentiation

- Research to speed up RSA, and the computing of elements in other cyclic groups of very large size -such as elliptic curves, and Fibonacci, or Lucas sequences- is usually pointed towards improving the multiplication.

- We aim to reduce the number of multiplications for a given fixed power, and series of powers.

In particular, we aim to optimise the final exponentiation in the pairing: $(p^k - 1)/r \in \mathbb{F}^*_{p^k}$

Contenido, sección 1

Intro

A fast method to compute $f^e \in \mathbb{F}^*_{p^k}$ is the square-and-multiply in which we can reuse intermediate values in the computation

Require: Positive integer e, $f \in \mathbb{F}^*_{p^k}$
Ensure: $f^e \in \mathbb{F}^*_{p^k}$

$g \leftarrow 1$
$\ell \leftarrow \lfloor log_2(e) \rfloor$
for $i = \ell - 1$ downto 0 **do**
  $g \leftarrow g^2$
  if $\ell_i = 1$ **then**
    $g \leftarrow g \cdot f$
  end if
end for
return g

Addition chains

A *probably* better approach is the use of addition chains:

Definición

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

Chain examples - 1

- Consider the following *Fibonacci* sequence: $\{1, 2, 3, 5, 8, 13, 21\}$. This is a addition chain for $e = 21$ which contains 7 elements. Each element is obtained from the addition of the previous two elements.
- An alternative chain is: $\{\ 1, 2, 4, 8, 16, 20, 21\ \}$. In this case, the element $e = 21$ can be constructed using 4 doubling operations and 2 addition operations, but has the same length.

Chain examples - 2

- Now, consider $e = 34$, the Fibonacci sequence grows by one element: $\{\ 1, 2, 3, 5, 8, 13, 21, 34\ \}$. We can also construct the following addition chain to reach 34: $\{\ 1, 2, 4, 8, 16, 32, 34\ \}$. This is a shorter chain and makes use of addition and doubling operations instead of only addition operations.

Chain examples - 3

- For the previous e examples, it is trivial to find a short addition chain with exhaustive search
- As e grows, the dificulty to determine if we have the shortest addition chain grows significantly (in deed, determining if we have the shortest chain is a NP-complete problem)

Addition Sequence

Addition Sequence

Given a list of integers $\Gamma = \{v_1, .., v_l\}$ where $v_l \geq v_i$ for all $i = 1, .., l-1$, 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 $e = v_l$.

Addition sequences, otherwise known as *multi-addition-chains*, are used to speed up the final exponentiation and for fast hashing to a point in G_2. To use these implementation improvements it is necessary to have code to generate the multi-addition chains for a given list of integers.

Note

- We refer to the set of integers which we wish to incorporate into an addition sequence as a “proto-sequence”.
- Some of its elements cannot be constructed by the addition of any other member of the set.

Contenido, sección 2

Introduction

First solutions

Our solution

Appendix - Code construction

Solution methods

Different methods exist to construct addition chains

- Bos and Coster presented a set of algorithms to construct addition chains.
- Bernstein presented a method for multi-scalar multiplication which constructs short addition sequences without the use of the Bos and Coster heuristic methods.
- Cruz-Cortés et al. presented a new approach to find short addition chains using Artificial Immune Systems
- Dominguez Perez y Scott extended Cruz-Cortez et al. method to addition sequences.

Bos and Coster

- Bos and Coster, suggested that an addition chain computation for an RSA exponent has to be fast to be useful, as one needs a different chain every time. In Pairing-Based Cryptography, this is not always true;

- They proposed a "Makesequence" algorithm. This algorithm starts with a proto-sequence 1, 2 and e, which we complete with at least one of the following methods:
  - Approximation
  - Division
  - Halving
  - Lucas.

Bernstein

- Another similar approach to the Bos and Coster method is to subtract elements from e. Bernstein presented a method for optimizing linear maps modulo 2, which incidentally, can be used to find a short addition chain. This is an example of the binary method.

- Instead of using subtractions it uses an in-place XOR with the two largest values in the chain (or a substraction), and repeating the operation until all of the elements are zero.

Artificial Intelligence (Heuristic)

- To automate the multi-addition chain code generation, we can use Artificial Intelligence to select which integers must remain in the sequence, and which can be discarded, this will continuously improve the sequence.

Contenido, sección 3

A new method

- The rationale behind Algorithm is to maximise the number of doubling steps associated to the output addition-subtraction sequence O to be produced, by processing separately the even and odd elements of the input set U in a backward fashion, *i.e.*, from the largest to the smallest element.

Essentially:

- Clasify even and odd elements
- Initialise with largest elements
- Main loop: *Valid*, and *invalid* elements
  - If invalid: include the half (or the difference with the closest element), and check for this element

Require: An ordered set of positive integers $U := \{e_1, e_2, \cdots, e_{s-1}, e_s\}$

Ensure: A valid addition-subtraction chain O for the input set U

$U_e := \{\forall\ e_i \in U | e_i\ mod\ 2 = 0\}$
$U_o := \{\forall\ e_i \in U | e_i\ mod\ 2 = 1\}$
$O := \emptyset;\ T_e := Max(U_e);\ T_o := Max(U_o);$
while $T_e \cup T_o \neq \emptyset$ **do**
  $\Delta = \emptyset$; $a_t := Max(T_e, T_o);$
  if IsNotValid$(a_t, U_e \cup U_o \cup O)$ **then**
    if IsEven(a_t) **then**
      $L := U_e \cup \{\forall\ e_i \in O | e_i\ mod\ 2 = 0\};$
    else
      $L := U_o;$
    end if
    for each $s \in L$ **do**
      $\Delta := \Delta \cup \{|a_t - s|\};$
    end for
    Lowest:=GetLowest(Δ);
    while IsEven(Lowest) and IsNotValid(Lowest,U_e $\cup U_o$ $\cup O$) **do**
      $O := O \cup$ {Lowest};
      Lowest := Lowest$/2$;
    end while
    if IsOdd(Lowest) **then**
      $U_o := U_o \cup$ {Lowest};
    else
      $O := O \cup$ {Lowest};
    end if
  end if
  if IsOdd(a_t) **then**
    $U_o := U_o - \{a_t\};$
  else
    $U_e := U_e - \{a_t\};$
  end if
  $O := O \cup \{a_t\};$
  $T_e := Max(U_e);\ T_o := Max(U_o);$
end while
O :=Sort(O);

Example

$$U = \{62, 87, 112, 248, 298\}$$

a_t	U_e	U_o	$Lowest$	T_e	T_o	O
-	$\{62, 112, 248, 298\}$	$\{87\}$	-	298	87	$\emptyset$
298	$\{62, 112, 248\}$	$\{25, 87\}$	50	248	87	$\{\underline{50}, 298\}$
248	$\{62, 112\}$	$\{17, 25, 87\}$	136	112	87	$\{\underline{34}, \underline{50}, \underline{68}, \underline{136}, 248, 298\}$
112	$\{62\}$	$\{17, 25, 87\}$	-	62	87	$\{\underline{34}, \underline{50}, \underline{68}, 112, \underline{136}, 248, 298\}$
87	$\{62\}$	$\{17, 25\}$	-	62	25	$\{\underline{34}, \underline{50}, \underline{68}, 87, 112, \underline{136}, 248, 298\}$
62	$\{\emptyset\}$	$\{3, 17, 25\}$	6	6	25	$\{\underline{6}, \underline{34}, \underline{50}, 62, \underline{68}, 87, 112, \underline{136}, 248, 298\}$
25	$\{\emptyset\}$	$\{1, 3, 17\}$	8	$\emptyset$	25	$\{\underline{2}, \underline{4}, \underline{6}, \underline{8}, \underline{25}, \underline{34}, \underline{50}, 62, \underline{68}, 87, 112, \underline{136}, 248, 298\}$
17	$\{\emptyset\}$	$\{1, 3\}$	16	$\emptyset$	17	$\{\underline{2}, \underline{4}, \underline{6}, \underline{8}, \underline{16}, \underline{17}, \underline{25}, \underline{34}, \underline{50}, 62, \underline{68}, 87, 112, \underline{136}, 248, 298\}$
3	$\{\emptyset\}$	$\{1\}$	2	$\emptyset$	3	$\{\underline{2}, \underline{3}, \underline{4}, \underline{6}, \underline{8}, \underline{16}, \underline{17}, \underline{25}, \underline{34}, \underline{50}, 62, \underline{68}, 87, 112, \underline{136}, 248, 298\}$
1	$\{\emptyset\}$	$\{\emptyset\}$	-	$\emptyset$	1	$\{\underline{1}, \underline{2}, \underline{3}, \underline{4}, \underline{6}, \underline{8}, \underline{16}, \underline{17}, \underline{25}, \underline{34}, \underline{50}, 62, \underline{68}, 87, 112, \underline{136}, 248, 298\}$

Comparison - 1

The Crypto'89 by Bos and Coster solution to $\{47, 117, 343, 499, 933, 5689\}$, is

$$\{\underline{1}, \underline{\mathbf{2}}, \underline{\mathbf{4}}, \underline{\mathbf{8}}, \underline{10}, \underline{11}, \underline{18}, \underline{\mathbf{36}}, 47, \underline{55}, \underline{91}, \underline{109}, 117, \underline{226}, 343, \underline{434}, \underline{489}, 499, 933, \underline{1422}, \underline{\mathbf{2844}}, \underline{\mathbf{5688}}, 5689\}, \tag{1}$$

whereas our Algorithm has the following solution:

$$\{\underline{1}, \underline{\mathbf{2}}, \underline{\mathbf{4}}, \underline{\mathbf{8}}, \underline{7}, \underline{\mathbf{16}}, \underline{\mathbf{32}}, \underline{39}, 47, \underline{63}, \underline{\mathbf{64}}, \underline{\mathbf{78}}, 117, \underline{\mathbf{126}}, \underline{\mathbf{128}}, \underline{\mathbf{156}}, \underline{\mathbf{256}}, \underline{217}, 343, \underline{\mathbf{434}}, 499, 933, \underline{1189}, \underline{\mathbf{2378}}, \underline{\mathbf{4756}}, 5689\}. \tag{2}$$

If $M = 3S$, the cost of Eq. (1) would be $16M + 6S = 54S$, whereas the cost of Eq. (2) would be $11M + 14S = 45S$.

Comparison - 2

- We used our algorithm to improve the final exponentiation in the KSS families of elliptic curves.
- Our results show a lower number of operations for the *hard part* of the final exponentiation in the Fuentes-Castaneda et al. method.

Curve	Benger	Fuentes-Castaneda *et al.*	This work
KSS-16	83M 20S	-	70M 14S
KSS-18	62M 14S	52M 8S	52M 6S
KSS-36	-	-	178M 68S

Contenido, sección 4

Vector Addition Chains

Definición

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 final exponent e.

Let V be a vector chain, and m be the dimension of every vector. The vector addition chain starts with $V_{i,i} = 1$ for $i = 0 \ldots m-1$: $[1, 0, 0, \ldots, 0]$,$[0, 1, 0, \ldots, 0]$,...,$[0, \ldots, 0, 1]$; we then proceed adding any two previous vectors to form a new vector in the chain, and continue until $V_{j,1} = e$ with j typically $> m$.

Olivos method

Given an addition chain $\Gamma = \{1, 2, 6, 12, 18, 30\}$ and its corresponding addition sequence $s = \{1, 2, \underline{3}, 6, 12, 18, 30, 36\}$, we construct a vector chain: $v = \{[1, 0], [0, 1], [1, 1], [2, 2], [3, 2], [6, 4], [12, 8], [18, 12], [30, 20], [36, 24]\}$.

	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	
t_0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	$\leftarrow$
t_1	2	2	0	1	0	0	0	0	0	0	0	0	0	0	$\leftarrow$
t_2	3	2	1	1	1	0	0	0	0	0	0	0	0	0	
t_3	6	4	2	2	2	2	1	0	1	0	0	0	0	0	$\leftarrow$
t_4	12	8	4	4	4	4	2	2	0	1	1	0	0	0	$\leftarrow$
t_5	18	12	6	6	6	6	3	2	1	1	0	1	1	0	$\leftarrow$
t_6	30	20	10	10	10	10	5	4	1	2	1	1	0	0	$\leftarrow$
t_7	36	24	12	12	12	12	6	4	2	2	0	2	2	2	$\leftarrow$

- To construct the vector chain matrix shown, we start with $[1, 0]$ and $[2, 2]$.
- To compute row $t_2 = [3, 2]$, we set $(t_0, c_0) = 1$ and $(t_1, c_1) = 1$ by induction (prioritizing a doubling over an addition).
- This means that $t_2 = t_0 + t_1$, which, translated into vector operations is equivalent to $[3, 2] = [1, 0] + [2, 2]$.
- The type of operation is expressed in $(t_2, c_2), (t_2, c_3)$ 1 denotes an addition, 2 denotes a doubling.

- The remaining cells of the column, if any, are the summation the corresponding cells in the column.
- For example, in $(t_5, c_6{-}\text{to}{-}c_7) = [3, 2]$ since $t_5 = t_4 + t_3$, hence $[3, 2] = [1, 0] + [2, 2]$.
- The arrows at the right of the table denote the rows containing the elements in Γ.

Conclusion

- 15