Number Theory & Cryptography - Weekly meetings

Curso en Zacatecas [Q1 2015]

Luis J. Dominguez Perez

CIMAT-Zac, Mayo 20 de 2015

Exponenciación

- La investigación para acelerar RSA, y el cómputo de elementos en otros grupos cíclicos de tamaño grande como curvas elípticas, y elementos en cadenas Fibonacci, o Lucas, normalmente se dirige a acelerar la velocidad de las multiplicaciones.

- En esta presentación se busca reducir el número de multiplicaciones para el cálculo de una potencia dada.

En particular, se busca optimizar la exponenciación a la potencia de $(p^k - 1)/r \in \mathbb{F}_{p^k}^*$, la cual se discutirá más adelante en el seminario.

Contenido, sección 1

Multiexponenciación

Intro

Un método más rápido para calcular $f^e \in \mathbb{F}_{p^k}^*$ que la exponenciación directa (square-and-multiply), puede estar basado en cadenas de adidión al reutilizar valores intermedios del cómputo

Definición

Una cadena de adición para un entero dado e es una secuencia $U = (u_0, u_1, u_2, \ldots, u_l)$ tal que $u_0 = 1$, $u_l = e$ y $u_k = u_i + u_j$ para $k \leq l$, y algún i, j con $0 \leq i \leq j$.

Ejemplos de cadenas

- 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.

Ejemplos de cadenas 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.

Ejemplos de cadenas 3

- Para los casos anteriores de e, es trivial encontrar una cadena de adición corta utilizando una búsqueda exhaustiva

- Conforme crece e, la dificultad de determinar la cadena de adición más corta crece considerable (de hecho, un problema NP-completo decidir si es la más corta)

Cadena de adición de vectores

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 $> m$.

Secuencia de adición

Secuencia de adición

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 be presented). 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.

Metodos para resolver

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-Cortez 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.

Método de 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.

Método de Bos and Coster

- **Approximation**: This method computes the difference between the target element and any two smaller elements already in the chain, adding the difference to the greatest of the smaller elements to include it in the chain.

- **Division**: inserts into the addition chain the set formed by $\lfloor e/S_i \rfloor$ for all of the elements in S, where S is a set of arbitrary length l containing the first l prime numbers.

Método de Bos and Coster

- **Halving**: inserts into the chain the set formed by $\{\lfloor L_i/a \rfloor\}$ for every element in L, where $a = e - s$, e is the target element, s is a random small element in the chain, and L is a set of arbitrary length containing small even integers.

- **Lucas**: include a Lucas sequence containing the target element e as the last element in the series.

Método de 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 XORs on the binary representation of the elements in the chain.

Método de Bernstein

- The algorithm applies an in-place XOR with the two largest values in the chain, and repeating the operation until all of the elements are zero.

- The addition chain is all of the values from the original chain plus the results of the XORs.

If the position of the most significant bit is not the same in both of the two largest elements, then we cannot reduce the element with an XOR. In this case, we have to use integer subtraction.

Construction of the Binary Method

In the Magma language there are no implementation of the binary XOR operation. However, one is able to obtain the binary representation of the coefficients in the addition chain with the use of the `Seqint` and `Intseq` functions for converting a sequence of "0"s and "1"s into its integer representation and vice-versa.

```
// Input  <- Ochain: proto-sequence to accomplish.
// Output -> Fchain: completed chain.
Reduce:=procedure(~Bchain,~a,~b,~Fchain)
  a,b:=GetHighest(Bchain); //a,b;
  if IsXORable(Bchain,a,b) then
    Bchain[a]:=BitXOR(Bchain,a,b);
  else
    tInteger:=Seqint(Bchain[a],2);
    tInteger-:=Seqint(Bchain[b],2);
    Bchain[a]:=Intseq(tInteger,2);
  end if;
  Include(~Fchain,Seqint(Bchain[a],2));
end procedure;

Bchain:=ToBitSeq(Ochain);
Fchain:=Ochain;
a:=0; b:=0;
```

```
while IsAllZero(Bchain) eq false do
  Reduce(~Bchain,~a,~b,~Fchain);
end while;
Exclude(~Fchain,0);
Sort(~Fchain);
Fchain;
```

Ejemplo

Consider the following proto-sequence:

$$\{1, 2, 4, 5, 7, 10, 15, 26, 28, 30, 55, 75, 80, 100, 108, 144\}.$$

Matrix

$$
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{matrix}
$$

Discusion

- Since the last row is equal to 144 and the second last is 108, the most significant bits are not both 1.
- We need to apply an integer subtraction, which in this case, replaces the last line with 36 $(0, 0, 1, 0, 0, 1)$
- Now, we focus our attention in 108 and 100 (second and third last rows respectively), which are the new highest values
- An XOR results in 8, which replaces the second to last line; then we continue with the rest

Discusion

The addition chain generated using this algorithm is:

$$\{1, 2, \underline{3}, 4, 5, \underline{6}, 7, \underline{8}, 10, \underline{11}, \underline{12}, 15, 26, \underline{27}, 28, 30, \underline{36}, 55, \underline{63}, 75, 80, 100, 1$$

which is of length 24 and requires 5 doubling and 19 addition operations.

Articifial Intelligence

- 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.

- We construct the code using vectorial addition chains

Metodo de Olivos

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.