Cryptography Based on the Linear Fibonacci Forms

Dr. Salem Sherif Elfard
Dept. of Computer Science
Faculty of Science- zawia
zawia University

Abstract:

The numbers play an important role in different theoretical and practical applications in the field of cryptography. Recently, it was discovered that they could be used successfully in accelerating arithmetic operations over large numbers. The practical impact of this information in various cryptography, which are known as heavily backed by large integer arithmetic. Cryptography is the only practical means to provide security services and it is becoming a powerful tool in many applications for information security.

The purpose of this paper is to give a parallel algorithm based on the linear Fibonacci forms method suggested by A.V. Anisimov, and applying it on a modular reduction on addition machines and also a modular exponentiation based on linear Fibonacci forms.

Key Words:

Addition Machines, Cryptography, Fibonacci numbers, linear combination, large numbers, Modular Exponentiation, Modular Reduction, Parallel Algorithm.

1. Introduction:

The reader is assumed to be familiar with basic notations of cryptography and cryptanalysis, for details the reader is referred to [1].

- Cryptographic Authentication; is the process of providing proof of identity of the sender to the recipient; so that the recipient can be assured that the person sending the information is who and what he or she claims to be [2].
- Linear Fibonacci forms have been introduced in [3]. The linear Fibonacci form of the range *t* is defined as a liner combination of the form :$xf_{t-1} + yf_t$,

  where x and y are integers and $y \neq 0$.

We call the linear from $xf_{t-1} + yf_t$, to be positively defined as a linear Fibonacci form only if it is deferent form the null and the coefficients *x* and *y* are natural numbers.

A representation of any natural numbers n exists as a positively defined linear Fibonacci form. For instance any trivial partition, $n = x + y = xf_1 + xf_2$ could be considered as the representation of the range 2. The representation of the maximum range is of interest to us because in such representation there is a possibility to maximally use the optimization effects of Fibonacci numbers.

2. Run-time Of The Euclidean Algorithm:

The Euclidean Algorithm ;is an algorithm that is largely used within cryptography and security in both Computer Science and the cryptography.

Given two natural numbers, the algorithm computes the greatest common divisor of the two numbers. When analyzing the run-time of the Euclidean Algorithm, it has been shown to have a worst case run-time of O(n) [4] .

Interestingly this proof also shows that the inputs that require the largest

number of computation steps, are inputs where the two numbers are successive Fibonacci numbers.

3. Modular reduction on addition machines:

An addition machine is a computation device with a finite number of registers, limited to the following six types of operations:

Read x *{input to register x}*

$x \leftarrow y$	*{copy register y to register x}*
$x \leftarrow x - y$	*{subtract register y from register x }*
$x \leftarrow x+y$	*{add register y to register x}*
$if\ x \geq y$	*{compare register x to register y}*
$Write\ x$	*{output from register x}*

The register contents are assumed to belong to a given set A, which is an additive subgroup of the real numbers. If A is the set of all integers, we say that the device is a real addition machine.

It was shown in [3] that the following evolutions of the running time in terms of addition machine operations are true (table 1).

Table 1 Time estimates for running on Fibonacci machines

Operation	**Running rime**
Remainders x mod y	$O(\log(x/y))$
Multiplication x . y	$O(\log \min(\lvert x\rvert, \lvert y\rvert))$
Integer part $\lfloor y/z \rfloor$	$O(log\left\lvert\frac{y}{z}\right\rvert)$
Greatest common deviser gcd (x,y)	$O(\log\ (\max\ (x, y)/\gcd(x, y))$
Exponentiation x^y mod z	$O((\log y)(\log z) + \log(x/z))$

In order to represent the main idea of the use of Fibonacci numbers let us consider the remainder function.

If we assume that two registers of an addition machine contain the pair of numbers $(y^{f_l}, y^{f_{l+1}})$, where l is an implicit parameter, it is easy to implement the operations $l \leftarrow 1,\ l \leftarrow l+1,\ l \leftarrow l-1$

And to test the conditions $x \geq F_l, x < yF_{l+1}, \ l = 1$.

Therefore *x mod y* can be computed efficiently by implementing the following procedure:

```
Read x; read y; { assume that x ≥ 0 and y > 0 }
  if $x \geq y$ then
  {
  $l \leftarrow 1$;
   repeat $l \leftarrow l + 1$ until $x < yF_{l+1}$;
  repeat if $x \geq yF_l$ then $x \leftarrow x - yF_t$;
   $l \leftarrow l - 1$;
  until $l = 1$;
 }
write x
```

The first repeat loop ascends l until we have $yF_t \leq x < yF_{l+1,}$ *i.e* $until\ l = \partial n$ where $n = \lfloor x/y \rfloor$, The second loop descends l while

$yF_{l_1} + yF_{l_2} + \ldots\ldots + yF_{l_n} = y_n$. Subtracting from x according to the Fibonacci representation of n . The result, $x - ny = x \bmod y$, has been computed with $2\partial n - 2 + vn = O(\log(x/y))$ using additions and subtractions altogether.

4. Modular Exponentiation based on Linear Fibonacci forms:

Theorem 1 [3] . If n – any natural number (not equal to null) then there exists the unique representation of this number as linear Fibonacci form, $n = af_{t-1} + bf_t$ At that inequalities hold true: $if\ a \neq 0$, then $0 < b < a < f_{t;}$ $if\ a = 0$ then $0 < b < F;\ a + b < c\sqrt{n}$; $(log_a n)/2 + c1 < t < log_a n$, $= (1 + \sqrt{5})/2, c1 > 0$, c and c_1 are small constants.

The proof is based on the idea of using sequentially two possible transformations which do not change the value of ***a*** integer. *If* $n = xf_{t-1} + y^f t$ and $y \geq x$ then it is possible to increase a current range,

$n = (y - x) f_t + xf_{t+1}$. If $y < x$ but $x \geq sf_t$ for some s; x and y can be reduced using the following transformation formula:

$$n\ (x - sf_{t-1})\ f_{t-1} + (y + sF_{t-1})F_{t}.$$

The purpose of using this transformation type is to reach, the first considered case if possible when the range is increased. The time for obtaining the linear Fibonacci form of the maximal range is *O(log n)* . It is interesting that there exists an algorithm on an addition machine under more restricted conditions constructing the maximal linear Fibonacci representation of the number *n* and using $O(log\ n)$ operations. If $n = af_{t-1} + bf_t$ is represented as the maximum linear Fibonacci form then numbers *a* and *b* also have the same representation. Corresponding coefficients for linear form of *a* or *b* have the similar representation in the same way and so on. In this manner we naturally get the notion of the linear Fibonacci tree for the number n.

4.1 The linear Fibonacci tree is constructed in the following way:

1. The starting vertex is marked by n.
2. If the formed vertex is marked by the number $z \neq 1$ then firstly we construct the linear Fibonacci representation for z for the maximum range,

   $z = xf_{t-1} + yf_t$ then we form two (or on the right *if* $x = 0$) vertexes – sons for the vertex z. The left edge of z is marked by f_{t-1} , where as the right edge marked by f_t . We mark the corresponding vertices of the lower level by x and y correspondingly.
3. Repeat step 2 on all vertices having marks distinguished form1.

Theorem 2 The depth of the linear Fibonacci tree of the number n is limited by the value $O(log\,log\,n)$.

Proof : The proof idea of this theorem consists of the following form the Theorem 1, one can get the fact that $n = af_{t-1} + bf_t$ is linear Fibonacci representation of the maximal range of numbers a and b are less than $c\sqrt{n}$ with the small constants c. Therefore, on the next level below for vertices marked by a and b the corresponding coefficients do not exceed $c\sqrt{c\sqrt{n}} = c^{1+1/2}\ n^{1/4}$ and so on. Taking consideration the sum *1 +1/2 + ¼ ...* is converged to 2 one can easily derive the theorem assertion.

The obtained above optimistic evolution displays that linear Fibonacci trees are good objects to represent large and even super large numbers.

Let us consider for instance the modular exponentiation problem x^y *mod z* it is evident that exponentiation into power equal some Fibonacci number is convenient and effective operation. Exponentiation x^{f1} *mod z* demands computation *x mod z* and *t* times using multiplication only on two registers. That is why we try to reduce the computation of x^y mod *z* to series of computation of the simple type u^{f_t} *mod z*.

If y= $af_{t-1} + bf_t$ is linear Fibonacci partition of *y* of the maximum range, then $x^y = (x^{f_{t-1}})^a * (x^{f_t})^b$ *mod z*.

Thus the problem is recursively reduced to a similar problem but with numbers of *O*(√ y) degree.

5. The linear Fibonacci tree is the base for constructing the parallel algorithm:

Input *x* is given to the root of the Fibonacci tree for *y*. The value $x^{f_{t-1}}$ mod *z* is transmitted down to the left son. Similarly the value x^{f_t} *mod z* is transmitted by the right channel. Computed results are transformed by the same way under decent on the next level. After reaching leaves the process changes its direction – results from vertices – sons are multiplied and transmitted up to a vertex – father. This algorithm demands *O(log log y)* passes with simple homogeneous computations on each level.

This algorithms demands *O(log y + log (x/z))* of working time in parallel implementation on computing architectures making linear modeling of binary trees.

In the above considered problem the computing tree is dynamically generated with data transmission along channels and with modifications of computing functions in vertices during running up and down. Such problems are well described in PARCS – technology of programming [5,6,7].

The given Recursive Procedure for exponentiation by mean of Linear Fibonacci forms will be as flows:

```
MPI < tre > powModFib ( MPKtre >  &  X, MPI < tre > &Y,
MPKtre > & M)
 {
 IF (X > = M ) return  powModFib (X.ModKhnut(M), Y, M):
 If (Y = 1) return X;
 If (Y = 2) return X. MulMod (X,M);
 If (Y is Zero) return 1;
 MPK tre> A,B,F1,F2,X1 = X, X2= X, f1, f2, Buf;
 Rang Fib (Y,A,B,F1,F2);
 f1 = 1, f2 = 1;
 While  (f2 < F2) {Buf = X2, X2 = X2. MulMod (X1,M),X1 = Buf, Buf
 =f2,
 f2 = f1, f1 = Buf;} if ( A . is Zero) return PowModFIB (X2,B,M);
 Return PowModFib (X1,A,M). MulMod (PowModFib (X2,B,M),M);
 }
```

Table 2 Computing x^y **mod M,** length of y = 62 bits. using Anisimov,s method;

Length of X and M in bits	Time in nsec	Length of X and M in bits	Time in nsec	Length of X and M in bits	Time in nsec
5	**0.55**	**165**	**203.99**	**326**	**857.08**
37	**10.93**	**197**	**295.,01**	**357**	**1041.8**
69	**37.51**	**229**	**412.89**	**389**	**1238.65**
101	**78.44**	**261**	**542.,51**		
133	**133.96**	**293**	**690.,65**		

Comparing with other methods, this method shows that it is good for small values of Y, but it is necessary to underline that recursive peculiarity of sequential realization can have an impact on time estimates. All experiments showed that in increasing the modulo M the difference in time was decreased. Theoretical considerations show that it should not be slower in methods that calculate very large powers.

6. Conclusion:

In this paper, theoretical and practical applications of Fibonacci numbers are presented, and used in the field of Cryptography Based on Linear Fibonacci Forms. A modular reduction on addition machines and a modular exponentiation based on Linear Fibonacci forms are discussed.

The paper explains the construction of linear Fibonacci tree. The conclusive part of the paper, as the recursive algorithm for exponentiation by mean of Linear Fibonacci forms, is given and its very perspective for parallel implementation.

References:

[1]M. Tahghighi, S. Turaev, R. Mahmod, A. Jafaar and M. Md. Said," The Cryptanalysis and Extension of the Generalized Golden Cryptography", IEEE conference on open system, September 2011, Lankawi, Malaysia.

[2]A. Joseph Raphael, Dr. V. Sundaram, "Secured Communication through Fibonacci Numbers and Unicode Symbols", International Journal of Scientific & Engineering Research, Volume 3, Issue 4, April-2012, ISSN 2229-5518.

[3] Anisimov A.V, "Linear Fibonacci Forms and Parallel Algorithms for High Dimension Arithmetic", Lecture Notes in Computer Science, Verlag 1995, pp 16-20.

[4]Phillip James, " When is a number Fibonacci?", Department of Computer Science, Swansea University, January 25, 2009

[5]Anisimov A.V, Kulyabko P.P. "Programming Parallel Computation in Controlling Space", \\ Cybernetic. 1984 No. 3. pp.79 -88.

[6]Kulyabko P.P, Anisimov A.V. "The peculiarities of PARCS technology of programming // Cebernetics and System Analysis", 1993 No.3, PP.128-137.

[7]Kulyabko P.P, Boreisha Y. E, "The Programming System PARCS // Programming", 1991. No.6, PP 91-102.