A public key cryptosystem based upon euclidean addition chains

Fabien Herbaut, Pascal Véron

A Public Key Cryptosystem Based upon Euclidean Addition Chains

Fabien Herbaut[1] and Pascal Véron[2]

[1] Université du Sud Toulon-Var, IMATH, France
IUFM de Nice, Université de Nice
herbaut@unice.fr
[2] Université du Sud Toulon-Var, IMATH, France
veron@univ-tln.fr

Abstract. Addition chains are classical tools used to speed up exponentiation in cryptographic algorithms. In this paper we proposed to use a subset of addition chains, the Euclidean addition chains, in order to define a new public key cryptosystem.

1 Introduction

The problem of minimizing the number of operations to compute x^n has a long history which involves al-Kashi and started at least in India, where the binary representation of n was already considered 200 B.C .

It appeared that this problem is deeply connected to this of finding short addition chains leading to n as explained in [6]. The name *addition chain* seems to come from Sholz paper [11].

Definition 1. *An addition chain of length s computing an integer k is a sequence $u_0, u_1, \ldots, u_s$ of positive integers such that :*

1. $u_0 = 1$ *and* $u_s = k$,
2. $\forall i \in [1, s]$, $u_i = u_j + u_t$ *with* $0 \leqslant j, t < i$.

Example : $(1, 2, 3, 6, 12, 15, 24, 39)$ is an addition chain of length 8 computing the integer 39, since $2 = 1 + 1$, $3 = 2 + 1$, $6 = 3 + 3$, $12 = 6 + 6$, $15 = 12 + 3$, $24 = 12 + 12$, $39 = 24 + 15$.

The problem of computing $l(n)$, the shortest length s of such a sequence computing n, is of importance and has given raise to numerous papers in the last century. For example, one can quote the papers of Brauer [2] , Yao [13], and the survey of Subbarao [12]. Two problems seem to have played the role of a red thread. The first one is to give sharp upper bounds for $l(n)$. As for example, it is well known that

$$\log n + \log v(n) - 2.13 \leqslant l(n) \leqslant \lfloor \log n \rfloor + v(n) - 1$$

where $v(n)$ is the Hamming weight of n. The Sholz conjecture, namely $\forall n \in \mathbb{N}^*, l(2^n - 1) \leqslant n - 1 + l(n)$, also played an important role in the development of the theory of addition chains.

C. Carlet and A. Pott (Eds.): SETA 2010, LNCS 6338, pp. 284–297, 2010.

The second problem is to find efficient algorithms to compute short chains for a given integer n. Both problems are still considered difficult. For recent results, one can see [1].

There is one special class of addition chains which have been well studied : the Brauer chains or star chains. This class is introduced in [2].

Definition 2. *A star addition chain or Brauer chain is a particular addition chain where* $\forall i \in [1, s]$, $u_i = u_{i-1} + u_j$ *with* $0 \leqslant j < i$.

EXAMPLE : $(1, 2, 3, 5, 8, 13, 26, 39)$ is a star addition chain of length 8 computing the integer 39.

These chains are well fitted for computations. Indeed at each step, to compute u_i, the last term u_{i-1} (already in the accumulator) is used. Recently, Meloni [8] studied a subclass of star chains : the so called Euclidean addition chains.

Definition 3. *An Euclidean addition chain (EAC) computing an integer* k *is an addition chain which satisfies* $u_1 = 1, u_2 = 2, u_3 = u_2 + u_1$ *and* $\forall\ 3 \leq i \leq s - 1$, *if* $u_i = u_{i-1} + u_j$ *for some* $j < i-1$, *then* $u_{i+1} = u_i + u_{i-1}$ *(case 1) or* $u_{i+1} = u_i + u_j$ *(case 2).*

As an EAC is a strictly increasing sequence, case 1 will be called big step (we add the biggest of the two possible numbers to u_i) and case 2 small step (we add the smallest one).

EXAMPLE : $(1, 2, 3, 4, 7, 11, 18, 25, 32, 39)$ is an Euclidean addition chain of length 10 computing the integer 39.

In [8], Meloni showed how to use such a chain (with a specific point addition algorithm) to compute nP where P is a point on an elliptic curve. Euclidean addition chains are also used in [5].

Computing an EAC for an integer n is easy : choose an integer $g < n$ such that $(g, n) = 1$ and apply Euclidean algorithm to n and g (see §2). In this way, one can find the $\varphi(n)$ EAC computing n (where φ is the Euler's totient function), but very few is known about the length of the chains obtained. A general asymptotic result due to Yao and Knuth [14] states that the average length of such a chain is

$$6\pi^{-2}(\ln n)^2 + \mathcal{O}(\log n(\log\log n)^2).$$

To find short EAC, Meloni suggests in [7] to choose g close to $\frac{n}{\phi}$ (where ϕ is the golden ratio) adapting this way a heuristic proposed by Montgomery [9] in the context of Lucas chains.

Nowadays, there are no known methods to find a chain of fixed length computing a prescribed integer n. The exhaustive method of listing the integers coprime with n and applying Euclidean algorithm will be clearly inefficient for large n as $\varphi(n)$ will be large too.

We will introduce in this paper a subset $\mathcal{M}_\ell^0$ of EAC of length 2ℓ such that two distinct elements of $\mathcal{M}_\ell^0$ will compute two different integers. Moreover, if $c \in \mathcal{M}_\ell^0$ computes an integer n, we will describe a simple and efficient method to determine c from the knowledge of n.

These remarks are our point of departure to propose a public key cryptosystem based upon EAC. Using chains of the set $\mathcal{M}_\ell^0$ induces a trapdoor in the problem of finding a chain of fixed length computing a prescribed integer.

This paper is organized as follows. Section 2 deals with links between Euclidean addition chains and the Euclidean algorithm. In section 3 we define the set $\mathcal{M}_\ell^0$ and give some of its properties. In section 4 we describe our cryptosystem. Section 5 deals with its security. We detail the scrambling actions of the cryptosystem, and show why they are important. We make links between difficult problems and the problem an intruder will have to solve to break the cryptosystem. We also discuss the parameters of the cryptosystem. In section 6 we discuss the performances of the cryptosystem. Section 7 gives a useful toy example which can help to better understand the cryptosystem. We conclude in section 8.

2 Euclidean Algorithm and Euclidean Addition Chains

For the sequel of the paper, we will use an equivalent definition for EAC. This way EAC can be in practice interpreted as binary sequences.

Definition 4. *An Euclidean addition chain (EAC) of length s is a sequence $(c_i)_{i=1\ldots s}$ with $c_i \in \{0,1\}$. The integer k computed from this sequence is obtained from the sequence $(v_i,u_i)_{i=0..s}$ such that $v_0 = 1$, $u_0 = 2$ and $\forall i \geqslant 1$, $(v_i,u_i) = (v_{i-1}, v_{i-1}+u_{i-1})$ if $c_i = 1$ (small step), or $(v_i,u_i) = (u_{i-1}, v_{i-1}+u_{i-1})$ if $c_i = 0$ (big step). The integer k associated to the sequence $(c_i)_{i=1\ldots s}$ is $v_s + u_s$.*

Example : From the EAC (1000111) one can compute the integer 39 as follows : $(1,2) \xrightarrow{1} (1,3) \xrightarrow{0} (3,4) \xrightarrow{0} (4,7) \xrightarrow{0} (7,11) \xrightarrow{1} (7,18) \xrightarrow{1} (7,25) \xrightarrow{1} (7,32)$, which corresponds to the EAC $1,2,3,4,7,11,18,25,32,39$.

From now on, we will define the length of an EAC as the length of the corresponding binary sequence $(c_i)_{i=1\ldots s}$.

Let us observe the progress of the substractive Euclidean algorithm when applied to coprime integers (see algorithm 1) in order to stress the link with EAC. The assertion $\{(v,u) = 1, v < u, u \geqslant 2, v \geqslant 1\}$ is an invariant of Algorithm 1. Moreover the variable u strictly decreases for each turn of the while loop. Hence the algorithm ends with $u = 2$ and $v = 1$.

Algorithm 1. Substractive Euclidean algorithm applied to coprime integers

Require: (v,u) with $(v,u) = 1$, $v < u$ and $v \geqslant 1$.
1: **while** $u > 2$ **do**
2: **if** $u \geqslant 2v$ **then**
3: $(v,u) \leftarrow (v, u-v)$
4: **else**
5: $(v,u) \leftarrow (u-v, v)$
6: **end if**
7: **end while**

Example : Starting from $(5, 17)$ the algorithm successively computes $(5, 12)$, $(\mathbf{5}, \mathbf{7})$, $(2, 5)$, $(\mathbf{2}, \mathbf{3})$ and $(1, 2)$ where bold couples mean that $u < 2v$. Now, if we read the sequence of the couples from the last one to the first one, notice that at each step the couple (v, u) is replaced by $(u, u + v)$ or by $(v, u + v)$. That is to say that reading the couples computing by Algorithm 1 from the last one to the first one we obtain an addition chain (as defined in definition 4) which can compute the initial input u.

Example : Starting from the previous example, we get $(1, 2) \xrightarrow{0} (\mathbf{2}, \mathbf{3}) \xrightarrow{1} (2, 5) \xrightarrow{0} (\mathbf{5}, \mathbf{7}) \xrightarrow{1} (5, 12)$, we obtain this way the EAC 0101 which computes the integer 17.

Taking into account this remark, we can easily define an algorithm computing an EAC for an integer k :

Algorithm 2. ComputeEACfor(k)

Require: $k \geqslant 4$.
1: Randomly computes an integer g, such that $g > k/2$ and $(g, k) = 1$.
2: $(v, u) \leftarrow (k - g, g)$
3: **while** $u > 2$ **do**
4: **if** $u \geqslant 2v$ **then**
5: $(v, u) \leftarrow (v, u - v)$
6: Output 1
7: **else**
8: $(v, u) \leftarrow (u - v, v)$
9: Output 0
10: **end if**
11: **end while**

Remark 1. Notice that in Algorithm 2, we choose $g > k/2$. Indeed suppose that $g \leqslant k/2$, then the first step of Algorithm 1 will compute the couple $(g, k - g)$ from (g, k). Now using the same algorithm with input (g', k) where $g' = k - g$, we will obtain after the first step the couple $(k - g', g') = (g, k - g)$ because $k - g \geqslant k/2$. Hence algorithm 2 applied to (g, k) or (g', k) will lead to the same EAC.

Notice also that, since $g > k/2$, the initialization $(v, u) \leftarrow (k - g, g)$ corresponds to the first execution of the While loop of Algorithm 1.

Remark 2. This algorithm outputs the mirror image of the EAC computing k when starting from an integer g (i.e. the sequence read from right to left). We will see in next section, that an EAC and its mirror image computes the same integer k.

3 Notations and Properties

We give in this section some notations and important results for the sequel of this paper.

Definition 5. *Let $n > 0$, we define :*

- *$\mathcal{M}$ as the set of EAC,*
- *$\mathcal{M}_n$ as the set of EAC of length $n > 0$,*
- *χ the map from $\mathcal{M}$ to $\mathbb{N}$, such that for $m \in \mathcal{M}$, $\chi(m)$ be the integer computed from the EAC m,*
- *ψ the map from $\mathcal{M}$ to $\mathbb{N} \times \mathbb{N}$, such that for $m \in \mathcal{M}$, $\psi(m) = (v_s, u_s)$ if $m \in \mathcal{M}_s$,*
- *S_0 the matrix $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ corresponding to a big step iteration,*
- *S_1 the matrix $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ corresponding to a small step iteration.*

With these notations, for $m = (m_1, \ldots, m_s) \in \mathcal{M}_s$, we have :

$$\psi(m) = (1,2)\prod_{i=1}^{s} S_{m_i} \text{ and } \chi(m) = \langle (1,2)\prod_{i=1}^{s} S_{m_i}, (1,1)\rangle.$$

Let r and s be two integers, we will denote by mm' the element of $\mathcal{M}_{r+s}$ obtained from the concatenation of $m \in \mathcal{M}_r$ and $m' \in \mathcal{M}_s$. This way, for $n > 0$, m^n is a word of $\mathcal{M}_{nr}$ if $m \in \mathcal{M}_r$.

Proposition 1. *Let $n > 0$, F_i be the i^{th} Fibonacci number (defined by $F_0 = 0$, $F_1 = 1$ and $F_{n+1} = F_n + F_{n-1}$) :*

- *$\psi(0^n) = (F_{n+2}, F_{n+3})$, $\psi(1^n) = (1, n+2)$, $\chi(0^n) = F_{n+4}$, $\chi(1^n) = n+3$,*
- *$\forall m \in \mathcal{M}_n$, $\chi(1^n) \leqslant \chi(m) \leqslant \chi(0^n)$,*
- *$S_0^n = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}$, $S_1^n = \begin{pmatrix} 1 & n \\ 0 & 1 \end{pmatrix}$.*

Proof. All these properties can easily be proved by induction.

Proposition 2. *Let $n > 0$ and $m = (m_1, \ldots, m_n) \in \mathcal{M}_n$, then :*

- *$\chi(m_1, \ldots, m_n) = \chi(m_n, \ldots, m_1)$,*
- *the map ψ is injective.*

Proof. We refer to [6] for standard link between EAC, Euclidean algorithm and continued fractions, which explains the first point. It is also explained that if $\psi(m) = (v, u)$ then $(u, v) = 1$ and the only chain which leads to (v, u) is obtained using the substractive version of Euclidean algorithm. □

From proposition 2 the restriction of χ to $\mathcal{M}_n$ is not injective because of the mirror symmetry property.

Proposition 3. *Let $\mathcal{M}_n^0$ be the subset of $\mathcal{M}_{2n}$ whose elements are EAC beginning with n zeros. The restriction of χ to $\mathcal{M}_n^0$ is injective.*

Proof. Let x and y be two words of $\mathcal{M}_n^0$ such that $\chi(x) = \chi(y)$, and $m0^n$, $m'0^n$, be the words obtained when reading x and y from right to left. Using the symmetry property, we have $\chi(m0^n) = \chi(m'0^n)$. Let $(v, u) = \psi(m)$ and $(v', u') = \psi(m')$, then

$$\begin{aligned} & \chi(m0^n) = \chi(m'0^n) \\ \Leftrightarrow\ & F_n u + F_{n-1} v + F_{n+1} u + F_n v = F_n u' + F_{n-1} v' + F_{n+1} u' + F_n v' \\ \Leftrightarrow\ & F_{n+2}(u - u') = F_{n+1}(v' - v)\,. \end{aligned}$$

Since $(F_{n+1}, F_{n+2}) = 1$, then F_{n+2} divides $v' - v$. Now from proposition 1, since v and v' are less or equal than F_{n+2} and nonzero, then $|v' - v| < F_{n+2}$. It implies that $v = v'$ and so $u = u'$. Hence $\psi(m) = \psi(m')$, so $m = m'$.

Proposition 4. *Let $c_{g,k}$ be the EAC computing the integer k from the integer g using Algorithm 2 then, $c_{g,k}$ ends with n zeros if and only if the n^{th} couple computed by Algorithm 2 is equal to $(kF_{n+1} - gF_{n+2}, gF_{n+1} - kF_n)$ if n is even or $(gF_{n+2} - kF_{n+1}, kF_n - gF_{n+1})$ if n is odd.*

Proof. Let us suppose that $c_{g,k}$ ends with n zeros. It means that the n^{th} couple computed by Algorithm 2 is equal to $(k - g, g)S_0^{-n}$. Now since $F_{n-1}F_{n+1} - F_n^2 = (-1)^n$ (Cassini's identity), then $S_0^{-n} = (-1)^n \begin{pmatrix} F_{n+1} & -F_n \\ -F_n & F_{n-1} \end{pmatrix}$. Hence $(k - g, g)S_0^{-n} = ((-1)^n(kF_{n+1} - gF_{n+2}), (-1)^n(gF_{n+1} - kF_n))$.

The converse can be easily proved by induction. □

Corollary 1. *Let $c_{g,k}$ be the EAC computing the integer k from the integer g using Algorithm 2. The chain $c_{g,k}$ ends with n zeros if and only if :*

- $k\frac{F_{n+2}}{F_{n+3}} < g < k\frac{F_{n+1}}{F_{n+2}}$, *if n is even.*
- $k\frac{F_{n+1}}{F_{n+2}} < g < k\frac{F_{n+2}}{F_{n+3}}$, *if n is odd.*

Proof. Let us suppose that $c_{g,k}$ ends with n zeros. From the preceding proposition, the n^{th} couple computed by Algorithm 2 is $((-1)^n(kF_{n+1} - gF_{n+2}), (-1)^n(gF_{n+1} - kF_n))$ and satisfies $(-1)^n(kF_{n+1} - gF_{n+2}) < (-1)^n(gF_{n+1} - kF_n)$. Thus $(-1)^n k\frac{F_{n+2}}{F_{n+3}} < (-1)^n g$. Now taking into account only the $n-1$ first steps, we also must have $(-1)^{n-1} k\frac{F_{n+1}}{F_{n+2}} < (-1)^{n-1} g$.

An easy induction proves the converse. □

The previous result means that to find an EAC (ending with n zeros) which computes an integer k, algorithm 2 has to be run with an integer g lying in a specific interval $\mathcal{I}$. Let $k \in \chi(\mathcal{M}_0^n)$ and c_k be the element of $\mathcal{M}_0^n$ such that $\chi(c_k) = k$. Let $\tilde{c}_k$ be the mirror of c_k, then $\tilde{c}_k$ ends with n zeros. The size $\mathcal{S}$ of the interval $\mathcal{I}$ is $|k\frac{F_{n+1}}{F_{n+2}} - k\frac{F_{n+2}}{F_{n+3}}|$ which is equal to $\frac{k}{F_{n+2}F_{n+3}}$. If $k < F_{n+2}F_{n+3}$ then $\mathcal{S} < 1$, hence at most one integer lies in $\mathcal{I}$. Now since k has been computed from a chain beginning with n zeros, then there is exactly one element g in $\mathcal{I}$ which can compute $\tilde{c}_k$ from k using algorithm 3.

Algorithm 3. InverseChi(k, n) for $k \in \chi(\mathcal{M}_0^n)$

1: **if** n is even **then**
2: $g \leftarrow \lfloor k\frac{F_{n+1}}{F_{n+2}} \rfloor$
3: **else**
4: $g \leftarrow \lfloor k\frac{F_{n+2}}{F_{n+3}} \rfloor$
5: **end if**
6: $(v, u) \leftarrow (k - g, g)$
7: **while** $u > 2$ **do**
8: **if** $u \geqslant 2v$ **then**
9: $(v, u) \leftarrow (v, u - v)$
10: Output 1
11: **else**
12: $(v, u) \leftarrow (u - v, v)$
13: Output 0
14: **end if**
15: **end while**

Remark 3. Since $\tilde{c}_k$ ends with n zeros, we can begin the preceding algorithm with :

0: *Output n zeros*

and (using proposition 4) modify the line 6 as follows :

$$6: (v, u) \leftarrow ((-1)^n(kF_{n+1} - gF_{n+2}), (-1)^n(gF_{n+1} - kF_n)).$$

Remark 4. Let $0^n y$ be a chain computing the integer k. The algorithm was designed to compute the chain $\tilde{y}0^n$ where $\tilde{y}$ is the mirror of y. But because of the progress of the algorithm the chain is sent back from the left to the right. Hence the last n bits returned are exactly the word y.

4 The Cryptosystem

The cryptosystem is composed of three algorithms :

- Genparam which takes as input two integers n and t $(n > t)$ and returns the public key pk and the secret key sk of the system,
- Encrypt which takes as input a binary sequence of size $n - t$, the public key pk and returns the cryptogram c,
- Decrypt which takes as input the cryptogram c, the secret key sk and return the plaintext m.

Let us give some details on the decryption procedure. To this end, we will denote by $\chi_{\alpha,\beta}(m)$ the integer computed from the EAC m when starting from the couple (α, β) instead of $(1, 2)$.

Let M be the matrix equal to $\prod_{i=1}^{n-t} S_{m_i}$ so that $\chi_{\alpha,\beta}(m) = \alpha(M_{11} + M_{12}) + \beta(M_{21} + M_{22})$. First notice that if d is the gcd of (α, β) then $\chi_{\alpha/d,\beta/d}(m) = \chi_{\alpha,\beta}(m)/d$, hence we will only consider the case where $\gcd(\alpha, \beta)$=1.

Algorithm 4. Genparam(n, t)

1: Randomly computes a prime $p > F_{2n+4}$
2: Randomly choose $\lambda \in [1, p-1]$
3: Randomly choose $x \in \{0,1\}^t$
4: $(\delta_1, \delta_2) \leftarrow \psi(0^n x) = \psi_{(F_{n+2}, F_{n+3})}(x)$
5: $(a, b) \leftarrow (\lambda\delta_1 \mod p, \lambda\delta_2 \mod p)$
6: $d \leftarrow \gcd(a, b)$
7: $pk \leftarrow (a/d, b/d)$
8: $sk \leftarrow (d, p, \lambda^{-1} \mod p, x)$
9: return (pk, sk)

Algorithm 5. Encrypt$(pk, m$: binary seq. of length $n-t)$

1: $c \leftarrow \chi_{pk}(m)$
2: return c

Algorithm 6. Decrypt(sk, c)

1: $y \leftarrow \lambda^{-1} dc \mod p$
2: $c_y \leftarrow$ InverseChi(y, n)
3: $m \leftarrow$ last $n-t$ bits of c_y (see Remark 4.).
4: return m

Let us notice in the same way $\psi_{\alpha,\beta}(m)$ the last couple obtained from the EAC m when starting from (α, β). Let m_1 and m_2 be any two EAC, then

- $\psi_{\alpha,\beta}(m_1 m_2) = \psi_{\psi_{\alpha,\beta}(m_1)}(m_2)$,
- $\chi_{\alpha,\beta}(m_1 m_2) = \chi_{\psi_{\alpha,\beta}(m_1)}(m_2)$.

Taking into account these results, we have the following equalities for the cryptosystem :

$$\chi(0^n xm) = \chi_{1,2}(0^n xm) = \chi_{F_{n+2},F_{n+3}}(xm) = \chi_{\delta_1,\delta_2}(m).$$

Now, since $c = \chi_{a/d,b/d}(m) = \chi_{a,b}(m)/d = \frac{a(M_{11}+M_{12})+b(M_{21}+M_{22})}{d}$, then

$$\lambda^{-1} cd \equiv \delta_1(M_{11} + M_{12}) + \delta_2(M_{21} + M_{22}) \mod p\,.$$

But,

$$\begin{aligned}\delta_1(M_{11} + M_{12}) + \delta_2(M_{21} + M_{22}) &= \chi_{\delta_1,\delta_2}(m) \\ &= \chi_{F_{n+2},F_{n+3}}(xm)\end{aligned}$$

and since $\chi_{F_{n+2},F_{n+3}}(xm) \leqslant \chi_{F_{n+2},F_{n+3}}(0^n) = F_{2n+4}$ (from property 2), then

$$\lambda^{-1} cd \mod p = \chi_{F_{n+2},F_{n+3}}(xm) = \chi(0^n xm)\,,$$

because $p > F_{2n+4}$. Using Algorithm 3, we can find back the sequence xm and deduce the plaintext m. Indeed, from a practical point of view, for the values n

suggested in section 6, $\chi(0^n xm) < F_{n+2}F_{n+3}$ as soon as the Hamming weight of x is greater or equal than 4. Another way to guarantee this last property is to consider only plaintext of length $n-1$. With such a condition, $\chi(0^n xm) \leqslant F_{2n+3} < F_{n+2}F_{n+3}$ for $n > 0$ and the map χ still remains injective. See section 7 for a toy example.

5 Security

First let us explain the meaning of the integer λ and the vector x. The integer λ is used in order to scramble the value of the couple (δ_1, δ_2). Indeed, if the cryptogram were computed as $\chi_{\delta_1,\delta_2}(m)$, then since $\chi_{\delta_1,\delta_2}(m) = \chi(0^n xm)$, any intruder could use Algorithm 3 to find back the cleartext m.

Remember that using x such that its Hamming weight be greater or equal than 4 guarantees that the value of $\chi(0^n xm)$ for any plaintext m is always strictly less than $F_{n+2}F_{n+3}$ (for the practical parameters given in section 6), which is an essential condition for the decryption process. Let us suppose however that we don't use the vector x, here is a possible attack to find back the secret parameters λ and p. Without x, (δ_1, δ_2) would be equal to (F_{n+2}, F_{n+3}). Now, if a and b are coprime, then pk will be equal to (a, b) in Algorithm 4. Hence, we will have

$$\begin{aligned} a &= \lambda F_{n+2} \mod p \\ b &= \lambda F_{n+3} \mod p \end{aligned}$$

i.e, there exist two integers j_a, j_b such that $a = \lambda F_{n+2} - j_a p$ and $b = \lambda F_{n+3} - j_b p$. Now, let $\varepsilon_0 = b$, $\varepsilon_1 = a$ and consider the sequence $\varepsilon_i = \varepsilon_{i-2} - \varepsilon_{i-1}$, a simple induction shows that $\varepsilon_i = \lambda F_{n+3-i} + (-1)^i (j_a F_i - j_b F_{i-1})p$, for $i \geqslant 2$. Hence $\varepsilon_{n+3} = (-1)^{n+3}(j_a F_{n+3} - j_b F_{n+2})p$ is a multiple of p. Since $F_k \mid F_{\ell k}$ we can obtain a set of integers which are all multiples of p. As an example since $F_4 = 3F_2$ and $F_{10} = 11F_5$, then $\varepsilon_{n-1} - 3\varepsilon_{n+1} \equiv 0 \mod p$ and $\varepsilon_{n-2} - 11\varepsilon_{n-7} \equiv 0 \mod p$. Computing the gcd of these integers will give us the value of p. Now, since $\varepsilon_{n+1} \equiv \lambda \mod p$ and $\lambda < p$, the value of ε_{n+1} modulo p gives us λ.

Using a vector x discards the possibility to easily obtain a set of multiples of p from the public key (a, b).

A way to find back the cleartext is to try to solve the following computational problem, which we will denote by **GEAC** for Generalized Euclidean Addition Chain Problem :

Name : GEAC
Input : Four integers a, b, α and ℓ such that $(a, b) = 1$ and $\alpha = \chi_{a,b}(c)$
Question : Compute $c \in \{0, 1\}^\ell$.

Suppose that an efficient algorithm could be designed to solve GEAC. If it is fast enough , it could then be used to compute minimal length EAC. As a consequence, using the method described in [8], this will lead to an efficient point multiplication algorithm for elliptic curves resistant to side channel attacks. From all the works done over addition chains, we did not find any references about the GEAC problem. Most of the papers on this topic deal with classical addition

chains starting with (1,2). It is thus of importance to classify this problem. We can associate a decision problem to GEAC :

Name : D-GEAC
Input : Four integers a, b, α and ℓ such that $(a, b) = 1$.
Question : Does there exist an euclidean addition chain c of length ℓ such that $\alpha = \chi_{a,b}(c)$?

We cannot state if this problem is NP-complete (it is clearly in NP). However, we would like to point out a related problem which is NP-complete, as we will prove it.

Name : G-AS
Input : A sequence $n_1, \ldots, n_r, a, b$ of positive integers such that $\gcd(a, b) = 1$, a positive integer L.
Question : Does there exist an addition chain of length $\leqslant L$ starting with (a, b) which contains all the $n_i's$?

This problem is a generalization of the following one :

Name : AS
Input : A sequence $n_1, \ldots, n_r$ of positive integers and a positive integer L.
Question : Does there exist an addition chain of length $\leqslant L$ which contains all the $n_i's$?

From [4] this problem is NP-complete.

Proposition 5. *G-AS is NP-complete*

Proof. The proof given in [4] shows how to reduce AS to the well known problem of Vertex Cover in a graph G. To this end, the author constructs the sequence $\Delta_G = \{1, 2, 2^2, \ldots, 2^{\sigma n}\} \cup \{1 + 2^{\sigma u} + 2^{\sigma v}\}$ where n is the number of vertices of G and (u, v) describes the set of edges. He shows then how to build a vertex cover of size at most K from an addition chain of size at most $\sigma n + 1 + \#E + K$ which contains the sequence Δ_G. Now, let us consider the sequence $\Delta_{Ga,b} = \{b, 2b, 2^2b, \ldots, 2^{\sigma n}b\} \cup \{a + b2^{\sigma u} + b2^{\sigma v}\}$ rather than Δ_G. Then we can read exactly the same proof to establish that G-AS is NP-complete. □

For a first approach of the security of the scheme, we must define parameters n and t in order to avoid classical attacks. The parameter t must be chosen so that an intruder cannot retrieve the chain x using an exhaustive search. We suggest to choose $t = 80$.

Since the size of the cleartext is $n-t$, we have to choose n such that $n-t > 80$, which leads to take $n > 160$.

The prime p must be chosen so that $p > F_{2n+4}$. We suggest to randomly select p in the range $]F_{2n+4}, F_{2n+5}]$. For $n > 160$, there are at least 2^{215} such primes.

Notice that since the cryptogram has been computed using the algorithm of definition 4 starting from $v_0 = a$ and $u_0 = b$ with $(a, b) = 1$ then all the couples (v, u) generated satisfy $(v, u) = 1$. Hence one could try to choose an integer $g < c$ coprime with c and apply algorithm 2 until the current couple (v, u) be equal to

(a, b). Now, there are about $\varphi(c)/2$ candidates and $\varphi(c) > c/\ln c$. Since c is of the order of p, selecting randomly g without any strategy will fail.

This cryptosystem is deterministic, and hence is not semantically secure, thus we do not resist to any of the **IND**-xxx attack. For this first approach of a cryptosystem based upon EAC, we do not investigate the formal model of provable security.

6 Performances

Let us first consider the transmission rate of this system. The size of the cleartext m is $n - t$. The cryptogram is obtained by the computation of

$$\langle (a, b) \prod_{i=1}^{n-t} S_{m_i}, (1, 1) \rangle .$$

If we consider the m_i's as $n - t$ independent Bernoulli random variables, it can be proved that the mean value of a cryptogram is $(3/2)^{n-t}(a + b)$. Since a and b are of the order of p, and since p is of the order of F_{2n+4}, this mean value is about $2(3/2)^{n-t}F_{2n+4}$. Taking into account that $\log_2 F_k$ is about $0.694k$, then the average size of the cryptogram is $1.97n - 0.58t + 3.7$. Hence the transmission rate of the cryptosystem is on average

$$\frac{n - t}{1.97n - 0.58t + 3.7} .$$

Since we fixed $t = 80$, and $n > 160$, this is an increasing sequence which tends to $1/1.96 \simeq 0.5$. Notice that the worst transmission rate is obtained when the cryptogram is computed from the cleartext 0^{n-t}. In this case the cryptogram is equal to $aF_{n-t+1} + bF_{n-t+2}$ whose size is about $2.08n - 0.69t + 4.16$.

The public and the private datas (except for x) are all of the order of p, which is close to F_{2n+4}. Using this estimation, table 1 sums up for $t = 80$ the characteristics of the system and give some numerical results for $n = 592$, $n = 1104$, $n = 2128$ and $n = 336$ (this last one is only given for illustrative purpose). The value $\mathcal{I}$ denotes the ratio between the size of the cleartext and the size of the cryptogram.The value $\mathcal{I}_W$ denotes the worst transmission rate.

Table 1. Characteristics of the scheme

n	size of cleartext (bits)	size of p_k (bits)	size of s_k (bits)	$\mathcal{I}$	$\mathcal{I}_W$
	$n - 80$	$2.8n + 5.6$	$4.2n + 88.4$	$\frac{n-80}{1.97n-42.83}$	$\frac{n-80}{2.08n-51.36}$
336	256	947	1500	0.41	0.39
592	512	1664	2575	0.45	0.43
1104	1024	3097	4726	0.48	0.46
2128	2048	5965	9026	0.49	0.47

The encryption process only involves $n-t$ additions over integers. The size of these integers grows from $1.4n$ (the size of a and b) to $2.08n$ in the worst case. We can speed up this process by using the following remark :

$$\chi_{pk}(m) = (a,b)\prod_{i=1}^{n-t} S_{m_i}(1,1)^t = (1,1)\prod_{i=n-t}^{1} S_{m_i}^t(a,b)^t\,.$$

Hence to cipher a cleartext m, the user can first compute $n-t$ additions between integers whose size grows from 1 to $0.69(n-t+2)$ in the worst case (the size of F_{n-t+2}). Then, he has to compute the products between integers of size about $1.4n$ and $0.7n$ (au and bv) and the sum $au+bv$.

The decryption process involves :

- step 1 of algorithm 6 : a modular multiplication between integers whose size is about $1.4n$, if we suppose that $\lambda^{-1}d$ has already been computed,
- step 2 or 4 of algorithm 3 : a multiplication between integers of size $1.4n$ and $0.694n$,
- step 2 or 4 of algorithm 3 : a division between an integer of size $2.1n$ and an integer of size $0.694n$,
- last steps of algorithm 3 : $n-t$ subtractions between integers whose size decreases from $1.4n$ to 1.

From an asymptotic point of view, both processes are in $\mathcal{O}(n^2)$ while the same procedures for the classical RSA cryptosystem are in $\mathcal{O}(n^3)$ if n is the size of the modulus. Table 2 gives some numerical results obtained when ciphering and deciphering 20000 cleartext with our cryptosystem and the classical RSA cryptosystem. Since in RSA the ciphering and deciphering procedure are identical we only mention in table 2 the time of ciphering procedure for a random exponent e. The column EAC* corresponds to the optimization of the encryption process above mentioned. Tests have been carried out on a Quadcore 2.33Ghz processor using GnuMP library.

Table 2. Ciphering and deciphering rate in kilobytes per second

size of the cleartext (bits)	EAC-cipher	EAC*-cipher	EAC decipher	RSA
1024	1106 kb/sec	2551 kb/sec	1208 kb/sec	103 kb/sec
2048	693 kb/sec	2024 kb/sec	963 kb/sec	28.46 kb/sec

The transmission rate of our system is a drawback of our system as compared to RSA. But since the design of this latter, very few new asymmetric cryptosystems have been proposed. For example, one could compare our parameters with those of another cryptosystem which didn't use an RSA-like mechanism : the Naccache-Stern knapsack cryptosystem [10] presented at Eurocrypt'97. We choose this cryptosystem since its parameters have been recently improved in 2008 [3]. Moreover, while the system lacks provable security, it still has not been

broken to this date. Since the encryption process involves modular multiplications and the decryption process is equivalent to an RSA signature, we will only discuss the transmission rate and the size of the public-key. In NS cryptosystem, there is a trade-off to establish between these two parameters. A good one corresponds to a transmission rate of 0.38 for a 512 kilobytes public key. If one wants to improve the transmission rate to 0.5, public key will grow up to 14564 kilobytes. On the other hand, for the smallest possible size of the public key (59 kilobytes), the transmission rate drops to 0.11. With our cryptosystem, for a transmission rate between 0.4 and 0.5, the public key is less than 1 kilobyte. Notice also that the proposed cryptosystem has a natural integrity property, since the cleartext computed from the cryptogram must be well formatted : the first $n+t$ bits should be equal to $0^n x$.

7 A Toy Example

We illustrate the mechanism for $n = 6$ and $t = 2$.

• Key generation
$p = 991 > F_{16}$, $\lambda = 230$, $x = (10)$
$(\delta_1, \delta_2) = (55, 76) = \psi(00000010)$ $((1,2) \xrightarrow{0} (2,3) \xrightarrow{0} (3,5) \xrightarrow{0} (5,8) \xrightarrow{0} (8,13) \xrightarrow{0} (13,21) \xrightarrow{0} (21,34) \xrightarrow{1} (21,55) \xrightarrow{0} (55,76))$
$(a, b) = (758, 633)$, $d = \gcd(a, b) = 1$
$pk = (758, 633)$, $sk = (1, 991, 642, (10))$ $(642 = 230^{-1} \mod 991)$.

• Encryption
Let $m = (1101)$ the message to encrypt, the following steps lead us to the computation of $\chi_{pk}(m)$:
$(758, 633) \xrightarrow{1} (758, 1391) \xrightarrow{1} (758, 2149) \xrightarrow{0} (2149, 2907) \xrightarrow{1} (2149, 5056)$
The cryptogram is 7205.

• Decryption
$y = \lambda^{-1} c \mod p = 7205 \times 642 \mod 991 = 613 < F_8 F_9 = 714$
$g = \lfloor 613 \frac{F_7}{F_8} \rfloor = 379$
Using the trick for the line 6 of algorithm 3, we initialize the couple (v, u) to $(613F_7 - 379F_8, 379F_7 - 613F_6) = (10, 23)$. Then the algorithm computes the following couples :
$(10, 13) \xrightarrow{1} (3, 10) \xrightarrow{0} (3, 7) \xrightarrow{1} (3, 4) \xrightarrow{1} (1, 3) \xrightarrow{0} (1, 2) \xrightarrow{1}$ end of algorithm. Last four bits are the cleartext m.

8 Conclusion

In this note we proposed to use Euclidean addition chains to define a public key cryptosystem. To this end, we used properties of a subset of Euclidean addition chains. It enabled us to design a polynomial time algorithm for the problem of

finding an EAC of fixed length computing a prescribed integer (GEAC). Even if we described difficult problems linked to GEAC, we do not know its level of difficulty. However, as we obtained good performances and as it is of interest to propose new public keys mechanisms, we think it is worth presenting this one. As it is usual in cryptography, we welcome readers for attacks and suggestions on this system. Although there exists a lot of efficient point multiplication algorithms for elliptic curves, few of them have been designed to intrisically resist to side channel attacks. Looking for an efficient cryptanalysis of GEAC may bring out new ideas in the theory of Euclidean addition chains. These ideas may have nice applications in the field of point multiplication algorithms resistant to side channel attacks.

References

1. Bahig, H.M.: Improved generation of minimal addition chains. Computing 78(2), 161–172 (2006)
2. Brauer, A.: On addition chains. Bull. Amer. Math. Soc. 45, 736–739 (1939)
3. Chevallier-Mames, B., Naccache, D., Stern, J.: Linear bandwidth naccache-stern encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 327–339. Springer, Heidelberg (2008)
4. Downey, P.J., Leong, B.L., Sethi, R.: Computing sequences with addition chains. SIAM J. Comput. 10(3), 638–646 (1981)
5. Goundar, R., Shiota, K., Toyonaga, M.: Spa resistant scalar multiplication using golden ration addition chain method. International Journal of Applied Mathematics 38(2), 83–88 (2008)
6. Knuth, D.E.: The Art of Computer Programming: Fundamental Algorithms, 3rd edn., vol. 2. Addison Wesley, Reading (July 1997)
7. Meloni, N.: Arithmétique pour la Cryptographie basée sur les Courbes Elliptiques. Ph.D. thesis, Université de Montpellier, France (2007)
8. Meloni, N.: New point addition formulae for ECC applications. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 189–201. Springer, Heidelberg (2007)
9. Montgomery, P.L.: Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains (2002), `ftp://ftp.cwi.nl/pub/pmontgom/Lucas.ps.gz`
10. Naccache, D., Stern, J.: A new public-key cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. Springer, Heidelberg (1997)
11. Sholz, A.: Aufgabe 253. Jahresbericht der deutschen Mathematiker-Vereinigung 47, 41–42 (1937)
12. Subbarao, M.: Addition chains - some results and problems. Number Theory and Applications, 555–574 (1989)
13. Yao, A.C.: On the evaluation of powers. SIAM Journal on Computing 5(1), 100–103 (1976)
14. Yao, A.C., Knuth, D.E.: Analysis of the substractive algorithm for greatest common divisors. Proc. Nat. Acad. Sci. USA 72(12), 4720–4722 (1975)