Published in *Electronics Letters* **32(6):537–538, 1996.**

Efficient computation of full Lucas sequences

M. Joye and J.-J. Quisquater

Indexing terms: Number theory, Cryptography

Recently, Yen and Laih (1995) proposed an algorithm to compute LUC digital signatures quickly. This signature is based on a special type of the Lucas sequence, V_k. The authors generalise their method to any type of Lucas sequence, and extend it to the 'sister' Lucas sequence, U_k. As an application, the order of an elliptic curve over $GF(2^m)$ is computed quickly.

Basic facts: In this section, we only include the minimal amount of background necessary to understand the Letter. For a systematic treatment, see [2, 3].

Let P and Q be two rational integers, and let α be a root of $x^2 - Px + Q = 0$ in the field $\mathbb{Q}(\sqrt{D})$, where $D = P^2 - 4Q$ is a non-square. Let β be the conjugate of α, *i.e.* $\beta = \bar{\alpha}$. The *Lucas sequences* $\{U_k\}_{k\geq 0}$ and $\{V_k\}_{k\geq 0}$ with parameters P and Q are given by

$$U_k(P,Q) = \frac{\alpha^k - \beta^k}{\alpha - \beta} \ , \tag{1}$$

$$V_k(P,Q) = \alpha^k + \beta^k \ . \tag{2}$$

It can easily be shown that the numbers U_i and V_i satisfy the following relations:

$$U_{i+j} = U_i V_j - Q^j U_{i-j} \ , \tag{3}$$

$$V_{i+j} = V_i V_j - Q^j V_{i-j} \ . \tag{4}$$

Fast algorithm for Lucas sequences: Assume you have to compute $U_k(P,Q)$ and $V_k(P,Q)$. If we use the binary expansion of k, it can be expressed as $k = K_0$, where

$$K_j = \sum_{i=j}^{n-1} k_i 2^{i-j}, \quad k_i \in \{0,1\} \text{ and } k_{n-1} = 1 \ .$$

Hence,

$$K_{j-1} = k_{j-1} + 2K_j = (K_j + k_{j-1}) + K_j \ . \tag{5}$$

Using Eqs. (3) and (4), we obtain

$$U_{K_{j-1}} = U_{(K_j + k_{j-1})} V_{K_j} - Q^{K_j} U_{k_{j-1}} \ , \tag{6}$$

$$V_{K_{j-1}} = V_{(K_j + k_{j-1})} V_{K_j} - Q^{K_j} V_{k_{j-1}} \ . \tag{7}$$

At iteration j, let $(l_j, h_j) = (K_j, K_j + 1)$. From Eq. (7), we see that both V_{l_j} and V_{h_j} are needed to compute $V_{l_{j-1}}$, and so to compute V_k. This is not the case for U_k, as we shall see in the following theorem.

Theorem: If k is odd, then the computation of U_k does not require the computation of U_{l_j} $(j \geq 1)$.

Proof: Since k is odd (i.e. $k_0 = 1$), $U_k (= U_{l_0}) = U_{h_1} V_{l_1} - Q^{l_1}$. Thus, only the value of U_{h_1} is needed. We only need to show that the value of $U_{h_{j-1}}$ can be derived from U_{h_j}. By Eq. (5) and depending on the value of k_{j-1}, we have the following cases:

- if $k_{j-1} = 0$, then $(l_{j-1}, h_{j-1}) = (2l_j, l_j + h_j)$;
- if $k_{j-1} = 1$, then $(l_{j-1}, h_{j-1}) = (l_j + h_j, 2h_j)$.

Hence, if $k_{j-1} = 0$, then $h_{j-1} (= h_j + l_j = 2l_j + 1)$ is odd and $U_{h_{j-1}} = U_{h_j} V_{l_j} - Q^{l_j}$; otherwise, $h_{j-1} (= 2h_j)$ is even and $U_{h_{j-1}} = U_{h_j} V_{h_j}$. ∎

```
Inputs:  k = 2^s Σ_{i=s}^{n-1} k_i 2^{i-s}, (k_s = 1)
         P, Q
Outputs: (U_k, V_k)

U_h = 1; V_l = 2; V_h = P; Q_l = 1; Q_h = 1;
for j from n − 1 to s + 1 by −1
     Q_l = Q_l * Q_h;
     if k[j] == 1 then
          Q_h = Q_l * Q;
          U_h = U_h * V_h;
          V_l = V_h * V_l − P * Q_l;
          V_h = V_h * V_h − 2 * Q_h
     else
          Q_h = Q_l;
          U_h = U_h * V_l − Q_l;
          V_h = V_h * V_l − P * Q_l;
          V_l = V_l * V_l − 2 * Q_l
     fi
endfor
Q_l = Q_l * Q_h; Q_h = Q_l * Q;
U_h = U_h * V_l − Q_l;
V_l = V_h * V_l − P * Q_l;
Q_l = Q_l * Q_h;
for j from 1 to s
     U_h = U_h * V_l;
     V_l = V_l * V_l − 2 * Q_l;
     Q_l = Q_l * Q_l;
endfor

{U_k(P,Q) = U_h; V_k(P,Q) = V_l}
```

Fig. 1 *Algorithm to compute* (U_k, V_k)

Remarks: (i) The presented algorithm is a left-to-right scanning one. Similar to [1], it is also possible to develop a right-to-left scanning algorithm, but that requires more temporary memories.
(ii) An implementation with Pari-GP [4] is available at

`ftp://math.math.ucl.ac.be/pub/joye/pari/lucas2.gp`

Performances: The worst case of the algorithm appears when $s = 0$. Assume $s = 0$; then the computation of U_k and V_k requires $\frac{11n}{2}$ multiplications. Furthermore, only five temporary memories (with the same length as the output) are needed.

If we only want the value of V_k, the algorithm is the same as that presented on figure 1, except that we do not care of the U_h's. Thus, the computation of V_k requires $\frac{9n}{2}$ multiplications in the worst case with only four temporary memories.

Remarks: (i) Some applications use Lucas sequences with parameter $Q = \pm 1$. In that case, the computation of U_k and V_k requires less than $3n$ multiplications with three temporary memories.
(ii) If we want U_k and V_k modulo a number, the computation can be improved using the technique of the common-multiplicand [5].
(iii) Moreover, owing to the high regularity of the algorithm, it can be parallelized.

Applications: Lucas sequences have numerous applications in number theory. For example, the divisibility properties of the U_k's (see [3, pp. 54–59]) allows to test the primality of a number N for which the factorization of $N + 1$ is partially given. It is also possible to develop efficient (pseudo) primality tests [6].

In this Letter, we shall see a less-known application. In 1985, the theory of elliptic curves emerged for cryptographic purposes. Since many cryptographic protocols [7, 8] require the knowledge of the order of an elliptic curve over $GF(2^m)$, *i.e.* $\#E(GF(2^m))$, we shall see how it can be computed using Lucas sequences.

Let p be a prime and $q = p^r$. Consider the elliptic curve $E/GF(q)$ such that $E(GF(q))$ has order $\#E(GF(q)) = q+1-t$. Using Weil theorem, we have

$$\#E(GF(q^l)) = q^l + 1 - \alpha^l - \beta^l \ , \tag{8}$$

Let $E/GF(2^r)$ be the non-supersingular elliptic curve given by the Weierstraß equation

$$E : y^2 + xy = x^3 + a_2x^2 + a_6 \ .$$

Assume r is small, so $t = 2^r + 1 - \#E(GF(2^r))$ can be computed by exhaustion. Moreover, if m is a multiple of l, then the curve E can be viewed as an elliptic curve over $GF(2^m)$. Hence, if we put $l = m/r$, then by Eq. (8)

$$\#E(GF(2^m)) = 2^m + 1 - V_l(t, 2^r) \ ,$$

where $V_l(t, 2^r)$ is the l^{th} term of the Lucas sequence $\{V_k\}$ with parameters $P = t$ and $Q = 2^r$.

Acknowledgments: The authors are grateful to Daniel Bleichenbacher and to Richard Pinch for providing useful informations about Lucas-based cryptosystems.

24 January 1996

M. Joye (*UCL Crypto Group, Département de Mathématique (AGEL), Université catholique de Louvain, Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium*)

J.-J. Quisquater (*UCL Crypto Group, Département d'Électricité (DICE), Université catholique de Louvain, Place de Levant 3, B-1348 Louvain-la-Neuve, Belgium*)

E-mail: joye@agel.ucl.ac.be / jjq@dice.ucl.ac.be

References

1 YEN, S.-M., and LAIH, C.-S.: 'Fast algorithms for LUC digital signature computation', *IEE Proc.-Comput. Digit. Tech.*, 1995, **142**, (2), pp. 165–169

2 RIESEL, H.: 'Prime numbers and computers methods for factorization' *in* 'Progress in Mathematics' (Birkhäuser, 1985), Vol. 57

3 RIBENBOIM, P.: 'The little book of big primes' (Springer, 1991)

4 BATUT, C., BERNARDI, D., COHEN, H., and OLIVIER, M.: 'User's guide to PARI-GP', January 1995

5 YEN, S.-M., and LAIH, C.-S.: 'Common-multiplicand multiplication and its applications to public key cryptography', *Electron. Lett.*, 1993, **29**, (17), pp. 1583–1584

6 POMERANCE, C., SELFRIDGE, J.L., and WAGSTAFF, S.S., JR.: 'The pseudoprimes to $25 \cdot 10^9$', *Math. of Comp.*, 1980, **35**, (151), pp. 1003–1026

7 MENEZES, A.J.: 'Elliptic curve public key cryptosystems' (Kluwer Academic Publishers, 1993)

8 MENEZES, A., QU, M., and VANSTONE, S.: 'Draft of IEEE P1363, chapter 6', November 1995