An efficient probabilistic public-key cryptosystem over quadratic fields quotients

Guilhem Castagnos

DMI-XLIM, Université de Limoges,
123, avenue Albert Thomas, 87060 Limoges CEDEX, France
guilhem.castagnos@unilim.fr

Abstract. We present a new probabilistic cryptosystem working in quadratic fields quotients. Computation in such objects can be done efficiently with Lucas sequences which help to design a fast system. The security of the scheme is based on the LUC problem and its semantic security on a new decisional problem. This system appears to be an alternative to schemes based on the RSA primitive and has a full computational cost smaller than the El Gamal EC cryptosystem.

Keywords: Probabilistic encryption; Lucas sequences; LUC cryptosystem; Quadratic fields; Catalano *et al.* cryptosystem.

1 Introduction

In 2001, Catalano, Gennaro *et al.* have proposed an efficient probabilistic cryptosystem (cf. [3]) whose security has been proved equivalent to the security of the RSA cryptosystem (cf. [4]). This scheme, which works in the ring $\mathbb{Z}/n^2\mathbb{Z}$ where n is an RSA integer, is a fast variant of the homomorphic scheme designed by Paillier ([13]) in 1999. The Catalano *et al.* scheme has then been adapted in the group of points of an elliptic curve by Galindo, Martín *et al.* in 2003 ([8]) with the help of the KMOV primitive (cf. [10]). However, the points of the elliptic curve considered have coordinates in the ring $\mathbb{Z}/n^2\mathbb{Z}$, where n is an RSA integer and the security of this system is based on the difficulty of factoring n. For discrete logarithm based schemes, the use of elliptic curves helps to reduce the size of the base field. As a consequence, the schemes are still competitive even if the operations in the group of points of an elliptic curve have an heavy cost. In the Galindo *et al.* scheme the base ring is huge so the system is much slower than the original one. Moreover, its decryption cost prevents this system to be used in practice. However, with this system, the authors have proposed an alternative to the use of the RSA function in probabilistic encryption and the encryption phase of their system is faster than the El Gamal EC scheme.

In this paper, we expose another alternative to the use of the RSA function much faster than the Galindo *et al.* scheme. The key idea is to adapt the Catalano *et al.* scheme in a group simpler than the elliptic curve over $\mathbb{Z}/n^2\mathbb{Z}$ used by Galindo *et al.*: the group of finite points of a conic over $\mathbb{Z}/n^2\mathbb{Z}$. In fact, this kind of group can be view as a group of norm 1 quadratic integers modulo n^2.

Quotients of quadratic fields have been quite famous in primality proving (Lucas pseudoprimes, see [1]) and factoring (the $p + 1$ method of Williams, cf. [16]). In cryptography, these groups have not been largely used. Only two systems have been proposed by Smith, Lennon and Skinner in 1994: the LUC cryptosystem (see [14]) and a signature scheme (see [15]). All these papers use the mysterious language of

Lucas sequences. In Section 2 we define a group, denoted $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$, of norm 1 quadratic integers modulo an integer a. Then, in Section 3, we show how Lucas sequences help to compute exponentiation in the group $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$. Thanks to this conceptualisation, we can prove easily, in Section 4, some interesting properties of the LUC function used in the systems of Smith. Moreover, its similarity with the RSA function becomes obvious.

We briefly recall the Catalano *et al.* cryptosystem in Section 5, then, in the next section, we show how to adapt this scheme with the LUC function, to design a new probabilistic system. The complexity of the new cryptosystem is analysed in Section 7 and its security is discussed in Section 8. Finally, in Section 9 we compare the efficiency of the new system with other probabilistic algorithms to show its competitiveness.

2 Notations and definitions

In all this paper, a positive integer n will be an RSA integer if n is the product of two distinct odd primes p and q (for the security of our cryptographic application, we will also later require that p and q are large enough, such that the factorisation of n is infeasible in reasonable time, see Section 8 for more details).

If A is a ring, $A^\times$ will denote the multiplicative group of the invertible elements of A. Given an integer Δ which is not a square, $\mathcal{O}_\Delta$ will be the ring of integers of $\mathbb{Q}(\sqrt{\Delta})$. Let a be an integer prime to Δ; as $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ is a free module of rank two over $\mathbb{Z}/a\mathbb{Z}$, we can define the norm application, denoted $\mathrm{N}_{\mathcal{O}_\Delta/a\mathcal{O}_\Delta}$ (resp. the trace application, denoted $\mathrm{Tr}_{\mathcal{O}_\Delta/a\mathcal{O}_\Delta}$) from $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ to $\mathbb{Z}/a\mathbb{Z}$, by assigning to every α in $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ the determinant (resp. the trace) of the endomorphism of $(\mathbb{Z}/a\mathbb{Z})$-module $\beta \mapsto \alpha\beta$.

If $\mathrm{N}_{\mathcal{O}_\Delta}$ (resp. $\mathrm{Tr}_{\mathcal{O}_\Delta}$) is the usual norm (resp. trace) in $\mathcal{O}_\Delta$ over $\mathbb{Z}$ then we have the following relations:

$$\forall \alpha \in \mathcal{O}_\Delta, \mathrm{N}_{\mathcal{O}_\Delta/a\mathcal{O}_\Delta}(\Pi(\alpha)) \equiv \mathrm{N}_{\mathcal{O}_\Delta}(\alpha) \pmod{a},$$

$$\forall \alpha \in \mathcal{O}_\Delta, \mathrm{Tr}_{\mathcal{O}_\Delta/a\mathcal{O}_\Delta}(\Pi(\alpha)) \equiv \mathrm{Tr}_{\mathcal{O}_\Delta}(\alpha) \pmod{a},$$

in which Π denotes the canonical map $\mathcal{O}_\Delta \to \mathcal{O}_\Delta/a\mathcal{O}_\Delta$. Considering these relations, we will denote simply the norm and trace applications respectively N and Tr.

As the norm is an homomorphism from the group $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\times$ to $(\mathbb{Z}/a\mathbb{Z})^\times$, the set of norm 1 elements of $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\times$ is a multiplicative group, which will be denoted $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$. One can also define this group as the set of finite points of the conic of affine equation:

$$X^2 - \Delta Y^2 = 1,$$

over $\mathbb{Z}/a\mathbb{Z}$. This is a consequence of the following fact: if $\alpha \in (\mathcal{O}_\Delta/a\mathcal{O}_\Delta)$, there exists a pair (x, y) of elements of $\mathbb{Z}/a\mathbb{Z}$ such that $\alpha = x + y\sqrt{\Delta}$ and $\mathrm{N}(\alpha) = x^2 - \Delta y^2$.

If p is an odd prime and x an integer such that $p \nmid x$, we denote by $\left(\frac{x}{p}\right)$ the Legendre symbol defined by $\left(\frac{x}{p}\right) = 1$ (resp. $\left(\frac{x}{p}\right) = -1$) if x is a quadratic residue modulo p (resp. is a quadratic nonresidue modulo p). The order $\varphi_\Delta(a)$ of $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$ is given by the following proposition:

Proposition 1. *Let Δ be a non-square integer*

- *Let p be an odd prime such that $p \nmid \Delta$, for all integer $r \geq 1$, the order of the group $(\mathcal{O}_\Delta/p^r\mathcal{O}_\Delta)^\wedge$ is $\varphi_\Delta(p^r) := p^{r-1}\left(p - \left(\frac{\Delta}{p}\right)\right)$.*
- *Let a be an odd integer, $a > 2$, with prime decomposition $a = p_1^{r_1} \cdots p_l^{r_l}$ where the primes $p_1, p_2, \ldots, p_l$ are all distinct. If $\gcd(\Delta, a) = 1$, we have the following isomorphism:*

$$(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge \xrightarrow{\sim} (\mathcal{O}_\Delta/p_1^{r_1}\mathcal{O}_\Delta)^\wedge \times \ldots \times (\mathcal{O}_\Delta/p_l^{r_l}\mathcal{O}_\Delta)^\wedge .$$

As a consequence, the order of $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$ is $\varphi_\Delta(a) := \prod_{i=1}^{l} \varphi_\Delta(p_i^{r_i})$.

3 Lucas sequences

3.1 Definition and computation

Let P and Q be two integers such that $P^2 - 4Q$ is a non-square. Lucas sequences are given by two second-order linear recurrence relations: $\forall k \geqslant 1$,

$$U_{k+1}(P,Q) = PU_k(P,Q) - QU_{k-1}(P,Q), U_1(P,Q) = 1, U_0(P,Q) = 0,$$
$$V_{k+1}(P,Q) = PV_k(P,Q) - QV_{k-1}(P,Q), V_1(P,Q) = P, V_0(P,Q) = 2.$$

There are several algorithms that allow to compute Lucas sequences (see [9], for example). These algorithms are analogous to the "square and multiply" algorithm for common exponentiation. For our purpose, we will only need to compute terms of the sequence $(V_n(P,1))_{n\in\mathbb{N}}$. We recall the corresponding algorithm:

Algorithm 1 (Computation of $V_k(P,1)$)

```
Input: $P \in \mathbb{Z}$ such that $P^2 - 4$ is a non-square and $k > 0$ with binary expansion
    $k = 2^s \sum_{i=0}^{m-1} k_i 2^{m-1-i}$ and $k_0 = k_{m-1} = 1$.
Initialisation: $V_l := 2, V_h := P$.
For $i = 0, \ldots, m-2$, do
    If $k_i = 0$,
        then, $V_h := V_l V_h - P, V_l := V_l^2 - 2$,
        else, $V_l := V_l V_h - P, V_h := V_h^2 - 2$.
    End If
End For
$V_l := V_l V_h - P$.
For $i = 1, \ldots, s$, do
    $V_l := V_l^2 - 2$.
End For
Output: $V_l$ which equals $V_k(P,1)$.
```

For our cryptographic application, we will compute $V_k(P,1)$ modulo an integer a. This will take m multiplications and $(m-1)+s$ squares in $\mathbb{Z}/a\mathbb{Z}$. Note that Algorithm 1 is faster than the general algorithm that computes $V_k(P,Q)$ with $Q \neq 1$. If k is odd (it will be the case in our application) then only the first loop of the algorithm is executed. In this case, the number and the nature of operations are both independent of the bits of k, this algorithm is thus immune to Simple Power Analysis attacks.

3.2 Lucas sequences and exponentiation in $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$

Lucas sequences have been used to design the LUC cryptosystem (see [14]) and a digital signature scheme (see [15]). These schemes have been well studied (see [2, 11]). Except in [2], all these papers are exclusively formulated in obfuscated terms of relations verified by Lucas sequences. However it is possible to avoid the systematic use of these relations, by exploiting the link between Lucas sequences and generic exponentiation in $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$.

Let $\Delta := P^2 - 4Q$ be a non-square integer like in 3.1. It is well-known that Lucas sequences allow to compute exponentiation in the ring $\mathcal{O}_\Delta$: we denote by α the element $\frac{P+\sqrt{\Delta}}{2}$ of $\mathcal{O}_\Delta$. This element is one of the roots of the polynomial $X^2 - PX + Q$ of $\mathbb{Z}[X]$. We then have

$$\forall r \in \mathbb{N}, \alpha^r = \frac{V_r(P,Q) + U_r(P,Q)\sqrt{\Delta}}{2},$$

in the ring $\mathcal{O}_\Delta$.

One can generalized this result to compute powers of elements of the ring $\mathcal{O}_\Delta$ with arbitrary non-square Δ. For our purpose, we state this generalised result modulo $a\mathcal{O}_\Delta$, where a is an odd integer prime to Δ:

Lemma 2. *Let Δ be a non-square integer and a an odd integer prime to Δ. Let α be an element of $\mathcal{O}_\Delta$ and x, y two integers such that $\alpha \equiv x + y\sqrt{\Delta} \pmod{a\mathcal{O}_\Delta}$. $\forall r \in \mathbb{N}$, we have*

$$\alpha^r \equiv \frac{V_r(2x, \mathrm{N}(\alpha))}{2} + yU_r(2x, \mathrm{N}(\alpha))\sqrt{\Delta} \pmod{a\mathcal{O}_\Delta},$$
$$\mathrm{Tr}(\alpha^r) \equiv V_r(2x, \mathrm{N}(\alpha)) \pmod{a\mathcal{O}_\Delta}.$$

Proof. Let $P = 2x$ and $Q = \mathrm{N}(\alpha) = x^2 - \Delta y^2$. The integer Δ is a non-square, and so is $P^2 - 4Q$. The result is trivial for $r = 0$ and $r = 1$. For all integers $r \geq 1$, it is straightforward to see that

$$\alpha^{r+1} = P\alpha^r - Q\alpha^{r-1}.$$

The lemma follows by induction using this equality combined with the linear recurrence relations verified by Lucas sequences. □

In the following, we will only deal with elements α of $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$. As for all integers P, $V(P, \mathrm{N}(\alpha)) \equiv V(P, 1) \pmod a$, and the same relation holds for the sequence U, the second parameter of the Lucas sequences used until the end of this paper will equal 1. We will therefore note $V_n(P)$ (resp. $U_n(P)$) for $V_n(P,1)$ (resp. $U_n(P,1)$).

4 The LUC function

Let n be an RSA integer. The encryption function of the LUC public key system (cf. [14]) is an adaptation of the RSA function in the group $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ with a speed improvement. For a parameter e, this function does not compute a full exponentiation $\alpha \mapsto \alpha^e$ in $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ but assign to an $x \in \mathbb{Z}/n\mathbb{Z}$, the trace of the element α^e with $\alpha \in (\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ chosen such that $x = \mathrm{Tr}(\alpha)$ (this idea is very similar to XTR, cf. [12]). According to Lemma 2, this computation can be done with the single Lucas sequence V.

In this section, we recall some properties of the LUC function, discuss its security and compare it to the security of the RSA primitive.

Definition 3. *Let $n = pq$ be an RSA integer and let Λ_n be the set:*

$$\Lambda_n := \left\{x \in \mathbb{N}, x < n, \gcd(x^2 - 4, n) = 1\right\},$$

and let e be an integer prime to $(p^2 - 1)(q^2 - 1)$, we define the LUC_e *function:*

$$\mathrm{LUC}_e : \begin{cases} \Lambda_n \to & \Lambda_n \\ \;x \;\mapsto V_e(x) \bmod n \end{cases}$$

We next state a property of the LUC encryption function:

Lemma 4. *With the notations of the previous definition,* LUC_e *is a permutation of the set Λ_n.*

Proof. We first prove that LUC_e is well defined. Let x be an element of Λ_n and Δ a non-square integer such that $\Delta \equiv x^2 - 4 \pmod{n}$. We have $\gcd(\Delta, n) = 1$. Let $\alpha \in \mathcal{O}_\Delta$ such that $\alpha \equiv \frac{x+\sqrt{\Delta}}{2} \pmod{n\mathcal{O}_\Delta}$. Modulo n, α is a norm 1 element and satisfies the relation $\mathrm{LUC}_e(x) \equiv V_e(x) \equiv \mathrm{Tr}(\alpha^e) \pmod{n}$. As $\mathrm{N}(\alpha^e) \equiv 1 \pmod{n}$ and $\alpha^e \equiv \frac{V_e(x)+U_e(x)\sqrt{\Delta}}{2} \pmod{n\mathcal{O}_\Delta}$ (cf. Lemma 2), we have $4\mathrm{N}(\alpha^e) \equiv V_e(x)^2 - U_e(x)^2\Delta \equiv 4 \pmod{n}$. So $V_e(x) \in \Lambda_n$ if and only if $U_e(x)$ is prime to n.

The order of $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ is

$$\varphi_\Delta(n) = \left(p - \left(\frac{\Delta}{p}\right)\right)\left(q - \left(\frac{\Delta}{q}\right)\right),$$

so e is prime to $\varphi_\Delta(n)$ and the homomorphism $\alpha \mapsto \alpha^e$ is an automorphism of the group $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$. The inverse map is $\alpha \mapsto \alpha^d$ where d is an integer such that $d \equiv e^{-1} \pmod{\varphi_\Delta(n)}$, in fact, e is the public exponent in LUC and d the private one. By Lemma 2, we have the following relation:

$$\alpha^{ed} = (\alpha^e)^d \equiv \frac{V_d(V_e(x)) + U_e(x)U_d(V_e(x))\sqrt{\Delta}}{2} \pmod{n\mathcal{O}_\Delta}.$$

Since we also have $\alpha^{ed} \equiv \alpha \pmod{n\mathcal{O}_\Delta}$, we must have $U_e(x)U_d(V_e(x)) \equiv 1 \pmod{n}$ and $V_d(V_e(x)) \equiv x \pmod{n}$. From the first equality, we deduce that $U_e(x)$ is prime to n and from the second equality, we deduce that $\mathrm{LUC}_d \circ \mathrm{LUC}_e = \mathrm{Id}_{\Lambda_n}$. Since e and d play a symmetric role, LUC_e is indeed a permutation of Λ_n (in fact LUC_e is the encryption function of LUC and LUC_d the decryption one). □

Corollary 5 (of Lemma 4). *With the notations of the previous definition, the function* LUC_e *is a permutation of the set*

$$\Lambda'_n := \left\{x \in \mathbb{N}, 0 < x < n, \gcd(x^2 - 4, n) = 1, \gcd(x, n) = 1\right\}.$$

Proof. It is sufficient to show that LUC_e is a self map of $\{x, \gcd(x,n) \neq 1\}$. Modulo p we must prove that $\mathrm{LUC}_e(0) \equiv 0 \pmod p$. Thanks to the choice of e, e is odd and $V_e(0) \equiv \mathrm{Tr}((\sqrt{\Delta}/2)^e) \equiv 0 \pmod p$, with $\Delta \equiv -4 \pmod p$. □

In the end of this section we briefly discuss the relation between the problems of inverting the LUC function and the RSA function. Let's first fix the notations with a definition.

Definition 6. *Let $n = pq$ be an RSA integer and e an integer prime to $(p-1)(q-1)$, we define the* RSA_e *function:*

$$\mathrm{RSA}_e : \begin{cases} (\mathbb{Z}/n\mathbb{Z})^\times \to (\mathbb{Z}/n\mathbb{Z})^\times \\ \qquad x \quad\;\; \mapsto \quad\; x^e \end{cases}$$

It is well known that the RSA function is a permutation of $(\mathbb{Z}/n\mathbb{Z})^\times$. The RSA and LUC function can be inverted by anyone who knows the inverse of e modulo the order of the group considered. In the two cases, this is equivalent to the knowledge of the factorisation of n. In [2, Section 5], the relation between the problem of inverting RSA_e and LUC_e is studied. It is said that for any $c \in (\mathbb{Z}/n\mathbb{Z})^\times$, one can compute $(\mathrm{RSA}_e)^{-1}(c)$ if he knows $(\mathrm{LUC}_e)^{-1}(c + c^{-1})$, but, as noticed by the authors, this fact does not imply that LUC is stronger than RSA.

One can also interpret these problems as the problem of finding a root of a polynomial over $\mathbb{Z}/n\mathbb{Z}$. For RSA, we have to find a root of $X^e - c$ and for LUC, we have to find a root of $V_e(X) - c$. One can prove by induction that $V_e(X)$ is a polynomial of degree e. More precisely, we have in $\mathbb{Z}[X]$:

$$V_e(X) = \sum_{k=0}^{\lfloor e/2 \rfloor} (-1)^k \frac{e(e-k-1)!}{k!(e-2k)!} X^{e-2k} = X^e - eX^{e-2} + \frac{e(e-3)}{2} X^{e-4} + \dots$$

Hence, to solve the two problems, one has to find a root of a polynomial of the same degree e. A result of Coppersmith gives the roots smaller than $n^{1/d}$ of a polynomial of degree d over $\mathbb{Z}/n\mathbb{Z}$ in polynomial time (cf. [5]). This result seems to indicate that the complexity of the problem of finding a root of a polynomial P over $\mathbb{Z}/n\mathbb{Z}$ is related to the degree of P. From this point of view, it makes sense to believe that RSA and LUC with same parameter e achieve the same level of security.

5 The Catalano, Gennaro *et al.* cryptosystem

In 2001, Catalano, Gennaro *et al.* have introduced in [3] a probabilistic cryptosystem. Their scheme is apparently very similar to the Paillier's cryptosystem (see [13]). In fact, the Catalano's system can also be view as a probabilistic version of RSA. Let's briefly describe this system.

As usual $n = pq$ is an RSA integer. Let e be an integer prime to $\varphi(n) = (p-1)(q-1)$. The pair (n, e) is the public key. We denote by R_n the set

$$R_n = \{x \in \mathbb{N}, 0 < x < n, \gcd(x,n) = 1\}.$$

The cipher function used in [3] is

$$\mathcal{E}_e : \begin{cases} \mathbb{Z}/n\mathbb{Z} \times R_n \rightarrow & (\mathbb{Z}/n^2\mathbb{Z})^\times \\ \quad (m,r) & \mapsto (1+mn)r^e \end{cases}$$

where m is the plaintext and r a random element. Note that, in practice, as the factorisation of n is not known by the one who uses the encryption function, r is taken randomly in $\{1,\ldots,n-1\}$ and as n must be hard to factor, $r \in R_n$ with high probability.

It is straightforward to show that this function is bijective: one uses the facts that $c := (1+mn)r^e \equiv r^e \pmod{n}$ and that the random integers r are distinct modulo n. Then, as the RSA function is a permutation of R_n, there is a unique r such that $\mathcal{E}_e(m,r) = c$. The uniqueness of m follows easily from the fact that $r^e \in (\mathbb{Z}/n^2\mathbb{Z})^\times$.

Let d be an integer such that $ed \equiv 1 \pmod{\varphi(n)}$. This integer is the secret key. To decrypt $c \in (\mathbb{Z}/n^2\mathbb{Z})^\times$, one has to reduce c modulo n, make an RSA decryption to recover r and then compute $c/r^e \pmod{n^2}$ to recover $(1+mn)$ and finally m. Although it's not clearly mentioned in [3], one can use the Chinese Remainder Theorem to speed up the decryption phase.

It is obvious that if one can invert the RSA function, one can also invert the function $\mathcal{E}_e$. In [4], the authors prove that to invert the RSA function is polynomially equivalent to the inversion of the $\mathcal{E}_e$ function. It is also proved that the system is semantically secure if and only if given an arbitrary element $c \in (\mathbb{Z}/n^2\mathbb{Z})^\times$, it is difficult to say if there exists an element $r \in R_n$ such that $c \equiv r^e \pmod{n^2}$ (see [3, theorem 3.1]).

Note that the system is not homomorphic contrary to the Paillier's cryptosystem because the map $r \mapsto r^e$ from $(\mathbb{Z}/n\mathbb{Z})^\times$ to $(\mathbb{Z}/n^2\mathbb{Z})^\times$ is not an homomorphism, unless n divide e, and e has to be chosen small enough in order to increase the cryptosystem efficiency.

6 The new cryptosystem

The Catalano *et al.* cryptosystem described in Section 5 has been adapted in the group of points of an elliptic curve over $\mathbb{Z}/n^2\mathbb{Z}$ by Galindo, Martín *et al.* in [8]. Algorithm 1 makes exponentiation in the group $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ faster than in elliptic curve and Section 4 gives an efficient trapdoor permutation. With all these elements, it becomes natural to adapt the Catalano *et al.* cryptosystem in the group $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$. We use the LUC$_e$ function (cf. Definition 3) to generate the random elements. As these elements must be invertible and distinct modulo n, we use the set Λ'_n defined in Corollary 5.

Definition 7. *Let $n = pq$ an RSA integer. Let e be an integer prime to $(p^2-1)(q^2-1)$. Let define the set Ω_n:*

$$\Omega_n := \left\{x \in \mathbb{N}, 0 < x < n^2, \gcd(x^2-4,n) = 1, \gcd(x,n) = 1\right\}$$

The encryption function is:

$$\mathcal{E}'_e : \begin{cases} \mathbb{Z}/n\mathbb{Z} \times \Lambda'_n \rightarrow & \Omega_n \\ \quad (m,r) & \mapsto (1+n)^m V_e(r) \bmod n^2 \end{cases}$$

Theorem 8. *With the notations of the previous definition, the function $\mathcal{E}'_e$ is well-defined and bijective.*

Proof. First we prove that $\mathcal{E}'_e$ is well-defined. Let (m, r) be an element of $\mathbb{Z}/n\mathbb{Z} \times \Lambda'_n$, we want to show that $c := \mathcal{E}'_e(m, r) \in \Omega_n$. We have $(c \bmod n) = \mathrm{LUC}_e(r)$. Since $r \in \Lambda'_n$, by Corollary 5, $(c \bmod n)$ is also an element of Λ'_n, hence $\mathcal{E}'_e$ is well-defined.

Now, we prove that $\mathcal{E}'_e$ is bijective. As the sets $\mathbb{Z}/n\mathbb{Z} \times \Lambda'_n$ and Ω_n have the same numbers of elements, $n(p-3)(q-3)$, it is sufficient to prove that $\mathcal{E}'_e$ is into. Suppose that there exist two pairs $(m_i, r_i)_{i=1,2}$ of $\mathbb{Z}/n\mathbb{Z} \times \Lambda'_n$ such that $\mathcal{E}'_e(m_1, r_1) = \mathcal{E}'_e(m_2, r_2)$. By taking this equality modulo n, we found that $\mathrm{LUC}_e(r_1) = \mathrm{LUC}_e(r_2)$. As LUC_e is a permutation of Λ'_n , we must have $r_1 = r_2$. As we have chosen the set Λ'_n in order to have $V_e(r_1)$ prime to n, we can conclude that m_1 equals m_2. □

We now give the encryption algorithm, the public key is (n, e) with the notation of Definition 7.

Algorithm 2 (Encryption)

```
**Input:** $(n, e)$ *the public key,* $m \in \mathbb{Z}/n\mathbb{Z}$ *the plaintext.*
**Randomization :** *Take* $r$ *randomly in* $\{1, \ldots, n-1\} \setminus \{2, n-2\}$.
**Output:** $c := (1 + mn)V_e(r) \bmod n^2$ *the ciphertext.*
```

Note that if n is hard to factor, we will have $r \in \Lambda'_n$ with probability close to 1. We now describe the decryption algorithm using the Chinese Remainder Theorem. The private exponents are given by the vector

$$d := (d_{(p,-1)}, d_{(p,1)}, d_{(q,-1)}, d_{(q,1)}),$$

where $d_{(p,i)} \equiv e^{-1} \pmod{p - i}$ for $i = \pm 1$, and the same notations holds with the prime q.

Algorithm 3 (Decryption)

```
Input: $(p, q, d)$ the private key, $c$ the ciphertext.
Precomputation:
    $inv_q := p^{-1} \bmod q, inv_p := q^{-1} \bmod p$.
    $pre_{\mathrm{CRT}} := p^{-1} \bmod q$.
For $l \in \{p, q\}$, do
    $i := \left(\frac{c^2-4}{l}\right)$,
    $r_l := V_{d_{(l,i)}}(c) \bmod l$.
End For
$r := r_p + p(r_q - r_p)pre_{\mathrm{CRT}} \bmod pq$.
For $l \in \{p, q\}$, do
    $tmp := c/V_e(r) \bmod l^2$,
    $tmp := (tmp - 1)/l$,
    $m_p :\equiv tmp \times inv_l \bmod l$.
End For
Output: $m :\equiv m_p + p(m_q - m_p)pre_{\mathrm{CRT}} \pmod{pq}$ the plaintext corresponding to $c$.
```

Proof. We prove that Algorithm 3 outputs a valid plaintext m given a ciphertext c. The first part of the algorithm recover the random number $r \in \Lambda'_n$ such that $c = \mathcal{E}'_e(m, r)$. Let Δ be a non-square integer such that $\Delta \equiv r^2 - 4 \pmod{n^2}$ and $\alpha \in \mathcal{O}_\Delta$ such that $\alpha \equiv \frac{r+\sqrt{\Delta}}{2} \pmod{n^2}$. We have $\mathrm{N}(\alpha) \equiv 1 \pmod{n^2}$ and $V_e(r) \equiv \mathrm{Tr}(\alpha^e) \pmod{n^2}$. According to Lemma 2, we have $4\mathrm{N}(\alpha^e) \equiv V_e(r)^2 - \Delta U_e(r)^2 \equiv 4 \pmod{n^2}$, where the last congruence holds because $\mathrm{N}(\alpha^e) \equiv 1 \pmod{n^2}$. We thus have

$$\Delta \equiv \frac{V_e(r)^2 - 4}{U_e(r)^2} \pmod{n^2}.$$

In this equation, $U_e(r)$ is invertible because if $r \in \Lambda_n$, by definition, Δ is prime to n and by Lemma 4, $\gcd(V_e(r)^2 - 4, n) = 1$.

As a consequence of this relation,

$$\left(\frac{\Delta}{p}\right) = \left(\frac{V_e(r)^2 - 4}{p}\right) = \left(\frac{c^2 - 4}{p}\right).$$

With the value of this last Legendre symbol, Algorithm 3 selects the inverse d_p of e modulo $\varphi_\Delta(p)$ and recover $r_p := V_{d_p}(c) \bmod p = V_{d_p}(V_e(r)) \bmod p = r \bmod p$. The algorithm then retrieves $r_q := r \bmod q$. To recover $m \bmod p$ from $c \bmod p^2$, we need to know $r \bmod p^2$. From r_p and r_q we recover $r \bmod n$ with the help of the Chinese Remainder Theorem: $r := r_p + p(r_q - r_p)pre_{\mathrm{CRT}} \bmod pq$. As r has been chosen smaller than n, we actually recover the value of r in $\mathbb{Z}$, and the values of r modulo p^2 and q^2 follow. We have $c/V_e(r) \equiv (1 + (mq)p) \pmod{p^2}$. We can thus compute $mq \bmod p$ and then $m \bmod p$. The same is done modulo q and the value of m modulo n is retrieved by an other application of the Chinese Remainder Theorem. □

7 Complexity of the new cryptosystem

In this section, we give the complexity of the encryption and decryption algorithms exposed in Section 6.

7.1 Notations and definitions

For simplicity, in all the following we will not distinguish the cost of modular squares and the cost of general modular multiplications of distinct elements. We will also neglect the cost of modular addition and modular subtraction.

In order to compare the efficiency of the new system with the previous ones, we express the complexity of encryption and decryption algorithms in terms of operations modulo n. For this purpose, we have to estimate the cost of operations modulo n^2.

We will use the following notations: let l be a non-negative integer, we note $|l|$ the number of bits of l. We will also denote by M_l (resp. I_l) the computational cost of a multiplication (resp. the cost of an inversion) modulo l.

Let x and y be two elements of $\mathbb{Z}/n^2\mathbb{Z}$ given in radix n representation: $x = x_0 + x_1 n$ and $y = y_0 + y_1 n$ where the x_i and y_i are integers smaller than n. The product xy can be done in radix n representation as follows: First compute the representation of the product $z := x_0 y_0$. We denote it $z = z_0 + z_1 n$. The product xy is then given by

$$xy = z_0 + (z_1 + x_0 y_1 + x_1 y_0)n.$$

With this method, we show that M_{l^2} is smaller than $3M_l$. The same method leads to the estimation: $I_{l^2} < (3M_l + I_l)$. Moreover, I_l is generally considered to be $10M_l$ (by the use of Lehmer's method). So $I_{l^2} < 13M_l$.

7.2 Encryption algorithm

With all these estimations, the cost of the computation of $V_e(r) \bmod n^2$ by Algorithm 2 is $3(2|e| - 1)M_n$. The multiplication $(1 + mn)V_e(r) \bmod n^2$ needs only one multiplication modulo n so the total cost of Algorithm 2 is

$$(6|e| - 2)M_n.$$

7.3 Decryption algorithm

Firstly, the Algorithm 3 computes two Legendre symbols in $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$. These computations takes $\mathcal{O}(\log^2 p)$ time. Then the computation of $r \bmod p$ takes at most $(2|p| - 1)M_p$. With the estimation $|p| = |n|/2$ and $M_p = M_n/3$, the total cost of the first step (computation of $r \bmod p$ and $r \bmod q$) is

$$\frac{2}{3}\left(|n| - 1\right) M_n.$$

Then, the computation of r by the Chinese Remainder Theorem takes only M_n. In the second phase of the algorithm, the computation of $V_e(r) \bmod p^2$ takes $2(|e| - 1)M_{p^2}$. With the estimation $M_{p^2} = M_n$, we find that retrieving $m \bmod p$ and $m \bmod q$ takes

$$\left(4|e| + \frac{50}{3}\right) M_n.$$

After adding the final use of Chinese Remainder Theorem, we find that the total cost of Algorithm 2 is

$$\left(\frac{2}{3}|n| + 4|e| + 18\right) M_n.$$

8 Security of the new cryptosystem

In this section, we discuss the security and the semantic security of the cryptosystem exposed in Section 6.

The security of the new cryptosystem relies on the difficulty of inverting the $\mathcal{E}'_e$ function, more precisely, on the difficulty of the following problem:

Problem 9. Let n be an RSA integer and e an integer prime to $(p^2 - 1)(q^2 - 1)$, let c be an element of the set Ω_n (cf. Definition 7) find $m \in \mathbb{Z}/n\mathbb{Z}$ such that there exists $r \in \Lambda'_n$ (cf. Corollary 5) such that

$$c = (1 + mn)V_e(r) \bmod n^2.$$

One can solve this problem if he knows how to invert the LUC function with the same parameters n and e. We refer to the end of Section 4 to find a discussion about the difficulty of this problem compared to the inversion of the RSA function. Equivalence between the two problems is not known where as the one-wayness of the Catalano *et al.* scheme is proved equivalent to the one-wayness of the RSA scheme. Factoring n is believed to be the more efficient way to invert the LUC function and we think that it is also the only way to invert the $\mathcal{E}'_e$ function. As a consequence, the prime factors p and q have to be chosen large enough to prevent n to be factorised, *i. e.*, p and q must have at least 512 bits.

Concerning the semantic security of the new cryptosystem we have the following theorem:

Theorem 10. *The semantic security of the new cryptosystem relies on the difficulty of the following decisional problem: Let n be an RSA integer and e an integer prime to $(p^2-1)(q^2-1)$, given an element c of the set Ω_n, find if there exists $r \in \Lambda'_n$ such that $c = (V_e(r) \bmod n^2)$.*

One can make a proof of this theorem with a straightforward adaptation of the proof of [3, theorem 3.1], concerning the semantic security of the Catalano *et al.* scheme. The only known way to solve the decisional problem exposed in Theorem 10 is to solve its computational version *i. e.*, given $c \in \Omega_n$, to find an $r \in \Lambda'_n$ such that $c = (V_e(r) \bmod n^2)$. As shown in the end of Section 4, this problem requires to find a root smaller than n of a polynomial of degree e over $\mathbb{Z}/n^2\mathbb{Z}$. Coppersmith's result (cf. [5]) fails to provide a polynomial time solution for all $e > 2$.

9 Comparison of the new cryptosystem with the previous one

In this section, we compare the complexity of our system with other probabilistic schemes working in finite groups of modular integers in order to use the cost of modular operations as an accurate method of comparison. For all the cryptosystems, the computational cost is estimated with the method used in Section 7. We also used the notations introduced in that section. The unity of complexity is M_n, *i. e.*, the cost of a multiplication modulo n. We always use Chinese remaindering for decryption. We first compare the new system with the other variants of Catalano *et al.* scheme.

System	Catalano *et al.*	Galindo *et al.*	New scheme	El Gamal EC
Group	$\mathbb{Z}/n^2\mathbb{Z}$	$E/(\mathbb{Z}/n^2\mathbb{Z})$	$\mathbb{Z}/n^2\mathbb{Z}$	$E/(\mathbb{F}_p)$
Plaintext size	$\lvert n\rvert$	$\lvert n\rvert$	$\lvert n\rvert$	$2\lvert p\rvert$
Ciphertext size	$2\lvert n\rvert$	$3\lvert n\rvert$	$2\lvert n\rvert$	$4\lvert p\rvert$
Encryption	$\frac{9}{2}\lvert e\rvert + 1$	$36\lvert e\rvert + 3$	$6\lvert e\rvert - 2$	$40\lvert p\rvert + 13$
Decryption	$\frac{\lvert n\rvert}{2} + 3\lvert e\rvert + 19$	$\frac{20}{3}\lvert n\rvert + 36\lvert e\rvert + 24$	$\frac{2}{3}\lvert n\rvert + 4\lvert e\rvert + 18$	$20\lvert p\rvert + 13$

In order to fairly compare these results, we must select key sizes that provide the same level of security. The first three systems use an RSA integer n and a public key exponent e. Since the security of all these systems is based on the difficulty of factoring n, we have to use the same modulus size. We will take $|n| = 1024$. The public exponent e must have the same size in the Catalano *et al.* scheme and in the new system. In the Galindo *et al.*, one can use an exponent of size twice smaller (see [8, Section 4.2] for details). A 16 bits public exponent e is taken for the new cryptosystem (due to Algorithm 1 a special exponent with a low Hamming weight will not speed up the computation), the special exponent $2^{16}+1$ (resp. 2^4+1) is taken for the Catalano *et al.* scheme (resp. for the Galindo *et al.* scheme) and the form of the exponent is taken in consideration to determine the number of multiplications necessary for exponentiation. For El Gamal EC, the difficulty of discrete logarithm in elliptic curves makes a prime of 192 bits sufficient to achieve a level of security as strong as an 1024 bits RSA modulus (In the FIPS publication which describe signature standards [6], the recommended size for prime base fields is at least 192 bits for ECDSA, and in the change notice enclosed to this document, the modulus size for RSA is required to be at least 1024 bits). We summarize the computational costs and keys sizes of these cryptosystems with these key length values in the following table:

System	Catalano	Galindo	El Gamal EC	New scheme
Plaintext	1024	1024	384	1024
Ciphertext	2048	3072	384	2048
Encryption	52	125	225	94
Decryption	565	6952	113	765
Public key	1040	1032	1344	1040
Private key	1040	1032	768	1040

The use of the Lucas function add a low computational cost compared to the original system of Catalano *et al.*. To compare more precisely our scheme with the El Gamal EC scheme, we give the computational cost per bit in M_n/bit:

System	El Gamal EC	New scheme
Encryption	0.586	0.092
Decryption	0.294	0.747
Encryption + Decryption	0.889	0.839

The new system has an encryption computational cost drastically smaller than the one of the El Gamal scheme. The decryption is not very expensive compared to that system, and is very reasonable compared to the Galindo *et al.* cryptosystem. Moreover, a full encryption and decryption phase is faster in the new cryptosystem. As a result, if one wants to use a probabilistic scheme whose security is not based on the RSA function the new cryptosystem has to be taken in serious consideration.

10 Further developments

The group of quadratic integers of norm 1 modulo an RSA integer is a tool to construct a lot of interesting cryptosystems. One could construct a probabilistic cryptosystem which work entirely in the group $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ by adapting the Catalano *et al.* cryptosystem using two key elements: the group automorphism $\gamma \mapsto \gamma^e$ with e prime to the order of $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ and the n^{th} root of unity $1 + n\sqrt{\Delta}$ where Δ is determined in order to have γ of norm 1. This solution gives a less efficient cryptosystem. Indeed, to compute the exponentiation γ^e, one has to compute both Lucas sequences V_e and U_e with suitable parameters. However this adaptation is still faster than the elliptic curve variant of Galindo *et al.*.

The new cryptosystem proposed in Section 6 and the variant of the previous paragraph are not homomorphic. However, an homomorphic cryptosystem could be built in $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ by adapting Paillier's ideas (cf. [13]). The ciphertext c corresponding to a plaintext $m \in \mathbb{Z}/n\mathbb{Z}$ is an element of $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ with $c = (1 + n\sqrt{\Delta})^m\rho$ where ρ is a random n^{th} power. The tricky part is to find a way to generate ρ in order to design an homomorphic system (the discriminant Δ must be fixed in order to have all the ciphertext in the same group). One solution is to publish a n^{th} power β of high order and to generate random ρ by computing β^r where r is random. A second solution is to use the map $(\alpha/\overline{\alpha})^n$ from $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\times$ to $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ that generates all the n^{th} power of $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ (If the factorisation of n is unknown, one can generate easily a random element of $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\times$, but it is difficult to build directly an element of norm 1). Both solutions give a cryptosystem slower than the one proposed in Section 6 as we have to compute both Lucas sequences V_n and U_n. However this cryptosystem is four times faster than the adaptation of the Paillier cryptosystem proposed by Galbraith (see [7]) in the group of points of an elliptic curve over the ring $\mathbb{Z}/n^2\mathbb{Z}$.

11 Conclusion

We have shown how to use the LUC function to design a competitive probabilistic scheme. Although this function and its underlying structure have not often been used in cryptography, they provide a competitive alternative to the RSA function. Moreover, the security of this function is believed at least as strong as the security of the RSA function.

References

1. R. Baillie and S. Wagstaff Jr, Lucas pseudo primes, *Mathematics of Computation* **152** (35), (1980) 1391–1417.
2. Daniel Bleichenbacher, Wieb Bosma and Arjen K. Lenstra, Some remarks on Lucas-based cryptosystems, *Proc. of CRYPTO'95*, Lecture Notes in Computer Science **963**, Springer Verlag (1995) 386–396.
3. Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, Phong Q. Nguyen, Paillier's cryptosystem revisited. *CCS'01: Proceedings of the 8th ACM conference on Computer and Communications Security*, ACM Press (2001) 206–214.
4. Dario Catalano, Phong Q. Nguyen, and Jacques Stern, The Hardness of Hensel Lifting: The Case of RSA and Discrete Logarithm. *Proc. of ASIACRYPT'02*, Lecture Notes in Computer Science **2501**, Springer Verlag (2002) 299–310.

5. Don Coppersmith, Finding a small root of an univariate modular equation, *Proc. of EUROCRYPT'96*, Lecture Notes in Computer Science **1070**, Springer Verlag (1996) 155–165.
6. FIPS PUB 186−2, *Federal information processing standards publication*, National Institute of Standards and Technology (2000)
7. Steven D. Galbraith, Elliptic Curve Paillier Schemes, *Journal of Cryptology* **15** (2), Springer Verlag (2002) 129–138.
8. David Galindo, Sebastiá Martín, Paz Morillo and Jorge L. Villar, An Efficient Semantically Secure Elliptic Curve Cryptosystem Based on KMOV, *Proc. of WCC'03*, 213–221.
9. Marc Joye and Jean-Jacques Quisquater, Efficient computation of full Lucas sequences. *Electronics Letters* **32**, (1996) 537–538.
10. Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto and Scott A. Vanstone, New Public-Key Schemes Based on Elliptic Curves over the Ring $\mathbb{Z}/n\mathbb{Z}$, *Proc. of CRYPTO'91*, 252–266.
11. Chi-Sung Laih, Fu-Kuan Tu and Wen-Chung Tai, On the security of the Lucas function, *Information Processing Letters* **53**, (1995) 243–247.
12. Arjen K. Lenstra and Eric R. Verheul, The XTR Public Key System, *Proc. of CRYPTO'00*, Lecture Notes in Computer Science **1880**, Springer-Verlag (2000), 1–19.
13. Pascal Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, *Proc. of EUROCRYPT'99*, Lecture Notes in Computer Science **1592**, Springer Verlag (1999) 223–238.
14. Peter Smith and Michael J. J. Lennon, LUC: A new public key system, in: E.G. Douglas, editor, *Ninth IFIP Int. Symp. on Computer Security*, Elsevier Science Publishers (1994) 103–117.
15. Peter Smith and Christopher Skinner, A Public-Key Cryptosystem and a Digital Signature System Based on the Lucas Function Analogue to Discrete Logarithms, *Proc. of ASIACRYPT'94*, Lecture Notes in Computer Science **917**, Springer Verlag (1994) 357–364.
16. H.C. Williams, A $p+1$ method of factoring, *Mathematics of Computation* **159** (39), (1982) 225-234.