The Case of Small Prime Numbers Versus the Joye-Libert Cryptosystem

George Teşeleanu[1,2]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
`tgeorge@dcti.ro`

[2] Simion Stoilow Institute of Mathematics of the Romanian Academy
21 Calea Grivitei, Bucharest, Romania

Abstract. In this paper we study the effect of using small prime numbers within the Joye-Libert public key encryption scheme. We introduce two novel versions and prove their security. We further show how to choose the system's parameters such that the security results hold. Moreover, we provide a practical comparison between the cryptographic algorithms we introduced and the original Joye-Libert cryptosystem.

Keywords: public key encryption, gap 2^k-residuosity, provable security
MSC: 11T71, 94A60, 68P25, 11A07, 11A15

1 Introduction

The Joye-Libert cryptosystem was introduced in [7, 20] as a generalisation of the Goldwasser-Micali public key encryption scheme [18, 19]. The main advantage of the Joye-Libert scheme compared to Goldwasser-Micali is that the first mentioned supports a larger message space, and thus it considerably decreases the expansion of the ciphertext. Regarding security, the authors prove that inverting the encryption function is equivalent to breaking the quadratic residuosity problem modulo $n = pq$, where p and q are prime numbers. Another important security result is that the scheme is semantically secure if the gap 2^k-residue assumption modulo n holds. We underline that the Joye-Libert cryptosystem is partially homomorphic with respect to message addition and supports ciphertext randomization.

In this paper, we aim at finding a way to improve decryption times for the Joye-Libert scheme. Therefore, we introduce two variants of the system: an unbalanced version and a multiprime one. In the first version we show how to decrease the size of p, while keeping the system secure and implicitly decreasing the complexity of decryption. The only cryptosystem related to this version is called the unbalanced RSA [31]. Just like our version, the scope is to reduce p, while keeping the system secure.

In the multiprime version, we increase the number of factors while keeping the size of the modulus constant and we manage to prove that inverting encryption is equivalent to a vectorial form of the quadratic residuosity assumption. We can find only a related cryptosystem in the literature: the multiprime RSA [4, 30]. The philosophy behind the multiprime RSA is the same as ours: uses parallelism to speed up decryption. A bonus of the multiprime Joye-Libert is that we can also use multiple threads to improve encryption time, while in the case of RSA this is not possible.

In the final section of our paper, we analyze the complexity of the two novel variants. Then we compare the decryption time complexities for all versions of the Joye-Libert scheme. If parallelization is possible, then the multiprime variant is to be preferred since it has faster encryption and decryption. Otherwise, the unbalanced version should be used.

Previous work. Note that a preliminary version of this proposal was presented in [27].

Structure of the paper. In Section 2 we provide the notations used in our paper. Then we recall several definitions needed to describe our proposals. The original Joye-Libert scheme is detailed in Section 3. In Sections 4 and 5 we present two novel versions of the Joye-Libert scheme. A performance analysis of the Joye-Libert variants is provided in Section 6. Conclusions and open problems are given in Section 7.

2 Preliminaries

Notations. In this paper, λ represents a security parameter. By $|n|$ we denote the size of n in bits. The action of selecting a random element x from a sample space X is denoted by $x \overset{\$}{\leftarrow} X$. The assignment operator $x \leftarrow y$ initialises variable x with value y. Let E be an event, then $Pr[E]$ represents the likelihood of E occurring. Probabilistic polynomial-time algorithms are referred to as PPT algorithms. In this paper, the set of natural numbers $\{0, \dots, a-1\}$ is denoted by $[0, a)$. To simplify notations we denote the set $[0, a+1)$ by $[0, a]$. Multidimensional vectors $v = (v_0, \dots, v_{s-1})$ are represented as $v = \{v_i\}_{i \in [0,s)}$.

2.1 Computational Complexity

In this subsection we present the reader with computational complexities of multiplication, exponentiation and modular inverse. These complexities are needed in order to determine the efficiency of our proposed schemes. Note that the asymptotic values are given in Table 1 and are taken from [15, 28]. To simplify our presentation, we use the notation $M(\cdot)$ for the complexity of the multiplication algorithm. Note that while discussing the complexity of performing an exponentiation we assume that the exponent's length is k bits.

Operation	Complexity
Multiplication	$M(\mu) = \mathcal{O}(\mu \log \mu \log \log \mu)$
Exponentiation	$\mathcal{O}(kM(\mu))$
Modular inverse	$\mathcal{O}(\mu M(\mu))$

Table 1: Computational complexity for μ-bit numbers

2.2 Number Theoretic Prerequisites

The Joye-Libert encryption scheme is based on a generalisation of the Legendre symbol, namely the 2^k-th power residue symbol. Note that the Legendre symbol is obtained when $k = 1$ and, for simplicity, we further denote it by $J_p(a)$. We recall the definition of the 2^k-th power residue symbol and some of its properties as stated in [33].

Definition 1. *Let p be an odd prime such that $2^k | p-1$. Then the symbol*

$$J_{p,2^k}(a) = a^{\frac{p-1}{2^k}} \bmod p$$

is called the 2^k-th power residue symbol modulo p, where $a^{\frac{p-1}{2^k}} \in \mathcal{Z}_p$, where $\mathcal{Z}_p = \{-(p-1)/2, \ldots, -1, 0, 1, \ldots, (p-1)/2\}$.

Lemma 1. *The 2^k-th power residue symbol satisfies the following properties*

1. *If $a \equiv b \bmod p$, then $J_{p,2^k}(a) = J_{p,2^k}(b)$;*
2. *$J_{p,2^k}(a^{2^k}) = 1$;*
3. *$J_{p,2^k}(ab) = J_{p,2^k}(a) J_{p,2^k}(b) \bmod p$;*
4. *$J_{p,2^k}(1) = 1$ and $J_{p,2^k}(-1) = (-1)^{(p-1)/2^k}$.*

Let $n = p_1 \ldots p_t$. We further denote by $J_n(a) = J_{p_1}(a) \ldots J_{p_t}(a)$ the Jacobi symbol of an integer a modulo an integer n. J_n and QR_n denote the set of integers modulo n with Jacobi symbol 1 and, respectively, the set of quadratic residues modulo n.

2.3 Public Key Encryption

The three PPT algorithms specific to a *public key encryption* (PKE) scheme are: *Setup*, *Encrypt* and *Decrypt*. Given as input a security parameter, the *Setup* algorithm outputs the public key and the corresponding secret key. To encrypt a message, *Encrypt* also needs the public key as input in order to output the correlated ciphertext. To recover the original message the *Decrypt* algorithm takes as input the secret key and the ciphertext. Note that if decryption fails, *Decrypt* returns an invalidity symbol.

Definition 2 (Indistinguishability under Chosen Plaintext Attacks - IND-CPA). *The security model against chosen plaintext attacks for a PKE scheme is described using the following game:*

Setup(λ)*: In this phase, challenger* C *first computes the public key, while keeping the corresponding secret key to himself. Then* C *sends only the public key to adversary* A*.*

Query: Adversary A *chooses two equal length messages* m_0, m_1 *and sends them to* C*. After flipping a coin* $b \in \{0, 1\}$*, the challenger encrypts* m_b *and sends to* A *the resulting ciphertext.*

Guess: In this phase, the adversary outputs a guess $b' \in \{0, 1\}$*.* A *wins the security game if* $b' = b$*.*

The advantage of an adversary A *attacking a PKE scheme is defined as*

$$ADV_A^{\text{IND-CPA}}(\lambda) = |Pr[b = b'] - 1/2|$$

where the probability is computed over the random bits used by C *and* A*. A PKE scheme is* IND-CPA *secure if for any PPT adversary* A*, the advantage* $ADV_A^{\text{IND-CPA}}(\lambda)$ *is negligible.*

3 The Joye-Libert PKE scheme

The Joye-Libert encryption scheme was initially introduced in [20]. Its security was improved in [7]. The authors proved that inverting the encryption function is as hard as the quadratic residuosity assumption. Joye *et al.* also showed that in the standard model the IND-CPA security of the PKE is equivalent to the gap 2^k-residuosity assumption. We shortly present the algorithms of the Joye-Libert cryptosystem.

Setup(λ): Set an integer $k \geq 1$. Randomly generate two distinct large prime numbers p, q such that $|p| = |q| = \lambda$ and $p \equiv 1 \bmod 2^k$. Output the public key $pk = (n, y, k)$, where $n = pq$ and $y \in J_n \setminus QR_n$. The corresponding secret key is $sk = (p, q)$.

Encrypt(pk, m): To encrypt a message $m \in [0, 2^k)$, we choose $x \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ and compute $c \equiv y^m x^{2^k} \bmod n$. Output the ciphertext c.

Decrypt(sk, c): Compute the value $z \equiv J_{p,2^k}(c)$ and find m such that the relation $\left[J_{p,2^k}(y)\right]^m \equiv z \bmod p$ holds. Efficient methods to recover m can be found in a subsequent section.

4 The Unbalanced Joye-Libert PKE scheme

In the unbalanced Joye-Libert scheme we reduce the size of p (denoted by λ_p) while keeping the size of n constant (denoted by λ_n). This modification only impacts the description of the *Setup* algorithm, which we briefly describe below. Therefore, we have $\lambda_n = \lambda_p + \lambda_q$, where $\lambda_q = |q|$ and $\lambda_p \leq \lambda_q$. Note that when $\lambda_p = \lambda_q$ we obtain the Joye-Libert cryptosystem, which we further refer to as the balanced Joye-Libert scheme.

Setup(λ_p, λ_q): Set an integer $k \geq 1$. Randomly generate two distinct large prime numbers p, q such that $|p| = \lambda_p$, $|q| = \lambda_q$ and $p \equiv 1 \bmod 2^k$. Output the public key $pk = (n, y, k)$, where $n = pq$ and $y \in J_n \setminus QR_n$. The corresponding secret key is $sk = (p, q)$.

Remark 1. Modifying the size of p does not impact the security proofs discussed in [7,20]. Therefore, as long as factoring is hard, the unbalanced version is secure. We show how to choose λ_p such that factoring remains difficult in Section 6.

5 The Multiprime Joye-Libert PKE scheme

5.1 Description

We further describe the multiprime Joye-Libert encryption scheme. In this case we split up n into multiple primes. Therefore, we have that $\lambda_n = t\lambda_p + \lambda_q$. Note that when $\lambda_p = \lambda_q$ and $t = 1$ we obtain the original cryptosystem from [7, 20] and, moreover, when in addition we set $k = 1$ we obtain the Goldwasser-Micali cryptosystem [18]. Also, if we set $t = 1$ we obtain the unbalanced version.

Setup(λ_p, λ_q): Set an integer $k \geq 1$. Randomly generate $t + 1$ distinct large prime numbers $p_1, \ldots, p_t, q$ such that $|p_1| = \ldots = |p_t| = \lambda_p$, $|q| = \lambda_q$ and $p_1, \ldots, p_t \equiv 1 \bmod 2^k$. Let $n = p_1 \ldots p_t q$. For each $i \in [1, t]$ select $y_i \xleftarrow{\$} \mathbb{Z}_n^*$ such that

$$J_{p_i}(y_i) = J_q(y_i) = -1 \quad \text{and} \quad J_{p_j, 2^k}(y_i) = 1, \text{ where } j \neq i.$$

Output the public key $pk = (n, Y, k)$ and the corresponding secret key $sk = P$, where $Y = \{y_i\}_{i\in[1,t]}$ and $P = \{p_i\}_{i\in[1,t]}$.

Encrypt(pk, m): To encrypt message $m \in [0, 2^k t)$, first we divide it into t blocks $m = m_1 \| \ldots \| m_t$ such that for each $i \in [1, t]$ we have $|m_i| = k$. Then, we choose randomly $x \xleftarrow{\$} \mathbb{Z}_n^*$ and compute the value $c \equiv x^{2^k} \cdot \prod_{i=1}^{t} y_i^{m_i} \bmod n$. The output is ciphertext c.

Decrypt(sk, c): For each $i \in [1, t]$, compute m_i using Algorithm 1.

Algorithm 1: Basic decryption algorithm

```
  Input: The secret prime $p_i$, the value $y_i$ and the ciphertext $c$
  Output: The message block $m_i$
1 $m_i \leftarrow 0$, $B \leftarrow 1$
2 foreach $s \in [1, k+1)$ do
3     $z \leftarrow J_{p_i,2^s}(c)$
4     $t \leftarrow J_{p_i,2^s}(y_i)^{m_i} \bmod p_i$
5     if $t \neq z$ then
6         $m_i \leftarrow m_i + B$
7     $B \leftarrow 2B$
8 return $m_i$
```

Correctness. Let $m_i = \sum_{w=0}^{k-1} b_w 2^w$ be the binary expansion of block m_i. Note that

$$J_{p_i,2^s}(c) = J_{p_i,2^s}(x^{2^k} \cdot \prod_{v=1}^{t} y_v^{m_v}) = J_{p_i,2^s}(y_i^{m_i}) = J_{p_i,2^s}(y_i)^{\sum_{w=0}^{s-1} b_w 2^w}$$

since

1. $J_{p_i,2^s}(x^{2^k}) = 1$, where $1 \leq s \leq k$;
2. $J_{p_i,2^k}(y_j) = 1$, where $j \neq i$;
3. $\sum_{w=0}^{k-1} b_w 2^w = \left(\sum_{w=0}^{s-1} b_w 2^w\right) + 2^s \left(\sum_{w=s}^{k-1} b_w 2^{w-s}\right)$.

As a result, the message block m_i can be recovered bit by bit using p_i.

5.2 Security Analysis

In this section we first introduce a vectorial generalisation of the quadratic residuosity problem and prove that inverting the encryption function of our proposal is as hard as breaking this assumption. Then we generalise the gap 2^k-residuosity problem stated in [20] and prove the IND-CPA security of our proposal.

Before stating the security assumptions used to prove the security of our proposal, we first define the following sets

$$\begin{aligned} \mathbb{Z}_n(i) &= \{x \in \mathbb{Z}_n^* \mid J_{p_j,2^k}(x) = 1 \text{ for any } j \neq i\}, \\ QR_n(i) &= \{x \in \mathbb{Z}_n(i) \mid J_{p_i}(x) = 1 \text{ and } J_q(x) = 1\}, \\ J_n(i) &= \{x \in \mathbb{Z}_n(i) \mid J_{p_i}(x) J_q(x) = 1\}, \end{aligned}$$

where $n = p_1 \dots p_t q$ and $i, j \in [1, t]$.

Definition 3 (Vectorial Quadratic Residuosity - VQR). *Choose $t+1$ distinct large prime numbers $p_1, \dots, p_t, q$ such that $|p_1| = \dots = |p_t| = \lambda_p$, $|q| = \lambda_q$ and $p_1, \dots, p_t \equiv 1 \bmod 2^k$. Let $n = p_1 \dots p_t q$. Let A be a PPT algorithm that returns 1 on input $(x_1, \dots, x_t, n)$ if $x_i \in QR_n(i)$. We define the advantage*

$$\begin{aligned} ADV_A^{\text{VQR}}(\lambda_p, \lambda_q) = \Big| & Pr[A(x_1, \dots, x_t, n) = 1 \mid x_i \xleftarrow{\$} QR_n(i) \text{ for } i \in [1,t]\ ] \\ & - \ Pr[A(x_1, \dots, x_t, n) = 1 \mid x_i \xleftarrow{\$} J_n(i) \setminus QR_n(i) \text{ for } i \in [1,t]\ ] \Big| . \end{aligned}$$

The Vectorial Quadratic Residuosity assumption states that for any PPT algorithm A the advantage $ADV_A^{\text{VQR}}(\lambda_p, \lambda_q)$ is negligible.

Theorem 1. *Inverting the encryption function of the multiprime Joye-Libert PKE is intractable if the* VQR *assumption is intractable.*

Proof. Let's assume that there exists an adversary A such that given a ciphertext c, it recovers the message m. We construct a PPT algorithms that breaks the VQR assumption. More precisely, on input $(y_1, \dots, y_t)$ B computes

$c \equiv (y_1 \dots y_t)^{2^{k-1}} x^{2^k}$, where $x \xleftarrow{\$} \mathbb{Z}_n^*$. Remark that if $y_i \in J_n(i) \setminus QR_n(i)$, then c is an encryption of $m_{2^{k-1}} = 2^{k-1} \| \dots \| 2^{k-1}$. If $y_i \in QR_n(i)$, then $y_i \equiv w_i^2 \bmod n$, and thus $c \equiv (w_1 \dots w_t x)^{2^k} \bmod n$ is an encryption of $m_0 = 0$. According to the above arguments, on input $(y_1, \dots, y_t, c)$ algorithm A outputs either $m_{2^{k-1}}$ or m_0, and thus B outputs 0 or 1, respectively. Therefore, B breaks the VQR assumption with non-negligible probability. $\square$

Definition 4 (Vectorial Gap 2^k-Residuosity - VGR). *Choose $t+1$ distinct large prime numbers $p_1, \dots, p_t, q$ such that $|p_1| = \dots = |p_t| = \lambda_p$, $|q| = \lambda_q$ and $p_1, \dots, p_t \equiv 1 \bmod 2^k$. Let $n = p_1 \dots p_t q$. Let A be a PPT algorithm that returns 1 on input $(x_1, \dots, x_t, k, n)$ if $x_i \in J_n(i) \setminus QR_n(i)$. We define the advantage*

$$ADV_A^{\text{VGR}}(\lambda_p, \lambda_q) = \Big| Pr[A(x_1, \dots, x_t, k, n) = 1 \mid x_i \xleftarrow{\$} J_n(i) \setminus QR_n(i) \text{ for } i \in [1,t]\ ] \\ - \ Pr[A(x_1^{2^k}, \dots, x_t^{2^k}, k, n) = 1 \mid x_i \xleftarrow{\$} \mathbb{Z}_n^* \text{ for } i \in [1,t]\ ] \Big|.$$

The Vectorial Gap 2^k-Residuosity assumption states that for any PPT algorithm A, the advantage $ADV_A^{\text{VGR}}(\lambda_p, \lambda_q)$ is negligible.

Theorem 2. *The multiprime Joye-Libert PKE is* IND-CPA *secure if and only if the* VGR *assumption is intractable.*

Proof. The proof of the statement is obtained by simply replacing the distribution of the public key elements $(y_1, \dots, y_t)$. More precisely, we select randomly the y_i values from the multiplicative subgroup of 2^k residues modulo n instead of drawing them from the $J_n(i) \setminus QR_n(i)$ set. Under the VGR assumption, the adversary will not notice this change. Therefore, we removed any link between the ciphertext c and the message m, and thus the IND-CPA security follows. $\square$

5.3 Optimizations

Setup Optimization. When choosing the public key we have to meet certain restrictions. An effective way to accomplish this is to first randomly choose the values $y_{i,i} \xleftarrow{\$} \mathbb{Z}_{p_i}^* \setminus QR_{p_i}$, $y_{i,t+1} \xleftarrow{\$} \mathbb{Z}_q^* \setminus QR_q$ and $w_{i,j} \xleftarrow{\$} \mathbb{Z}_{p_j}^*$. Then compute the elements $y_{i,j} \leftarrow w_{i,j}^{2^k} \bmod p_j$. Using the Chinese remainder theorem we compute the desired value $y_i \in \mathbb{Z}_n^*$ such that $y_i \equiv y_{i,\ell} \bmod p_\ell$ for all $\ell \in [1,t]$ and $y_i \equiv y_{i,t+1} \bmod q$.

Decryption Optimization. In order to speed-up the decryption process, the authors of [21] add an extra restriction when generating the prime factors of n, and then use it to simplify decryption. Applying this to the multiprime case, we obtain the fact that we have to generate p_i such that $p_i \not\equiv 1 \bmod 2^{k+1}$ holds.

Let $p_i' = (p_i - 1)/2^k$ and $\alpha_i[s] = 2^{k-s}p_i'$. Then the following relation between the ciphertext and plaintext holds

$$\begin{aligned}
c^{\alpha_i[s]} &\equiv (x^{2^k} \cdot \prod_{v=1}^{t} y_v^{m_v})^{\alpha_i[s]} \\
&\equiv y_i^{\alpha_i[s] \sum_{w=0}^{s-1} b_w 2^w} \\
&\equiv y_i^{b_{s-1} 2^{k-1} p_i'} y_i^{\alpha_i[s] \sum_{w=0}^{s-2} b_w 2^w} \\
&\equiv (-1)^{b_{s-1}} y_i^{\alpha_i[s](m_i \bmod 2^{s-1})} \bmod p_i
\end{aligned}$$

since

1. $(x^{2^k})^{\alpha_i[s]} = x^{2^{k-s}(p_i-1)} = 1$;
2. $J_{p_i,2^k}(y_j) = 1$, where $j \neq i$;
3. $\sum_{w=0}^{k-1} b_w 2^w = \left(\sum_{w=0}^{s-1} b_w 2^w\right) + 2^s \left(\sum_{w=s}^{k-1} b_w 2^{w-s}\right)$;
4. $J_{p_i}(y_i) = -1$.

Hence, if we precompute $D_i = y_i^{-p_i'}$, then we can recover message block m_i. Wrapping it all together we obtain Algorithm 2. Note that when $t = 1$ we obtain [21, Algorithm 3]. The authors also propose three other optimizations[3] [21, Algorithm 4, 5 and 6], but their complexity is similar with Algorithm 3's complexity.

Algorithm 2: Optimized decryption algorithm

```
Input: The secret values $(p_i, D_i)$, the value $y_i$ and the ciphertext $c$
Output: The message block $m_i$
1  $m_i \leftarrow 0$, $B \leftarrow 1$, $D \leftarrow D_i$
2  $C \leftarrow J_{p_i,2^k}(c)$
3  foreach $j \in [1, k-1]$ do
4      $z \leftarrow C^{2^{k-j}} \bmod p_i$
5      if $z \neq 1$ then
6          $m_i \leftarrow m_i + B$
7          $C \leftarrow C \cdot D \bmod p_i$
8      $B \leftarrow 2B$, $D \leftarrow D^2 \bmod p_i$
9  if $C \neq 1$ then
10     $m_i \leftarrow m_i + B$
11 return $m_i$
```

[3] Note that two of these optimizations contain a typo: in line 5, Algorithm 5 and line 6, Algorithm 6 we should have $A^{k-j} \neq C[k-j] \bmod p$ instead of $A \neq C[k-j] \bmod p$.

6 Implementation and Performance Analysis

6.1 Parameter Selection

The fastest currently known algorithm for factoring composite numbers is the Number Field Sieve (NFS) [26]. The expected running time of the NFS depends on the size of the modulus n and not on the size of its factors. More precisely, the expected running time is approximately

$$L[n] = e^{1.923(\log n)^{1/3}(\log\log n)^{2/3}}.$$

In [25,26], the authors extrapolate the running time needed to factor a modulus of size λ_n from the computational effort required to factor a 512-bit modulus. Hence, a λ_n-bit modulus offers a security equivalent to a block cipher of d-bit security if

$$L[2^{\lambda_n}] \simeq 50 \cdot 2^{d-56} \cdot L[2^{512}]. \tag{1}$$

Since we start from a secure Joye-Libert PKE and we wish to optimize decryption by decreasing the size of some of the factors of the modulus, while keeping the size of the modulus constant, the NFS cannot be expected to factor n. Unfortunately, this strategy can make the resulting PKEs vulnerable to the Elliptic Curve Method (ECM) [22], if we lower the size of the factors below a certain threshold. Compared to the NFS, the ECM has the running time determined by the size of the smallest factor. Thus, if p is the smallest factor, then the running time of the ECM is

$$E[n,p] = (\log_2 n)^2 e^{\sqrt{2\log p \log\log p}}.$$

Similarly to the NFS, Lenstra [24] extrapolates the equivalent security provided by a module of size λ_n with the smallest prime of size λ_p to be

$$E[2^{\lambda_n}, 2^{\lambda_p}] \geq 80 \cdot 2^{d-56} \cdot E[2^{768}, 2^{167}]. \tag{2}$$

From Equations (1) and (2) we can deduce the following equivalency

$$E[2^{\lambda_n}, 2^{\lambda_p}] \geq 80 \cdot 2^{\log_2(L[2^{\lambda_n}]/(50\cdot L[2^{512}]))} \cdot E[2^{768}, 2^{167}]. \tag{3}$$

A different model for predicting the security against the NFS and the ECM is provided in [11]. Compared to Lenstra's model, Brent uses known historical factoring records to predict the year a modulus of a given size will be factored. Using the least-squares fit, Brent obtains the following equation for the NFS

$$D_n^{1/3} = \frac{Y - 1928.6}{13.24} \text{ or equivalently } Y = 13.24 \cdot D_n^{1/3} + 1928.6 \tag{4}$$

and for the ECM

$$D_p^{1/2} = \frac{Y - 1932.3}{9.3} \text{ or equivalently } Y = 9.3 \cdot D_p^{1/2} + 1932.3, \tag{5}$$

where D_n is the number of digits of the factored modulus and D_p is the number of digits of the largest prime factor found using the ECM.

Using regression analysis, we update Brent's equations using data points from [3,5,8,9,12,13,16,17,23,29,32] for the NFS and from [10,34] for the ECM. These data points are presented in Figures 1 and 2.

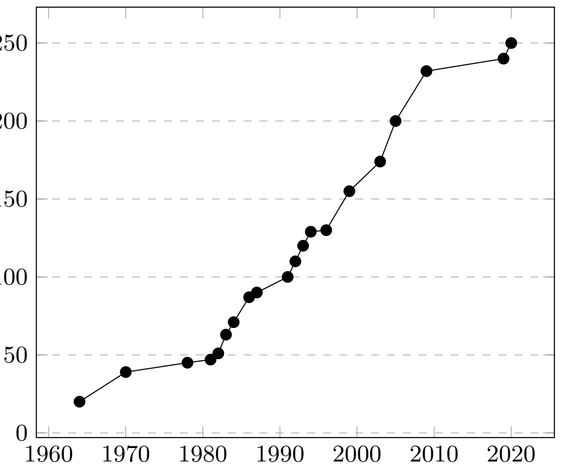


Fig. 1: Size of general number factored versus year

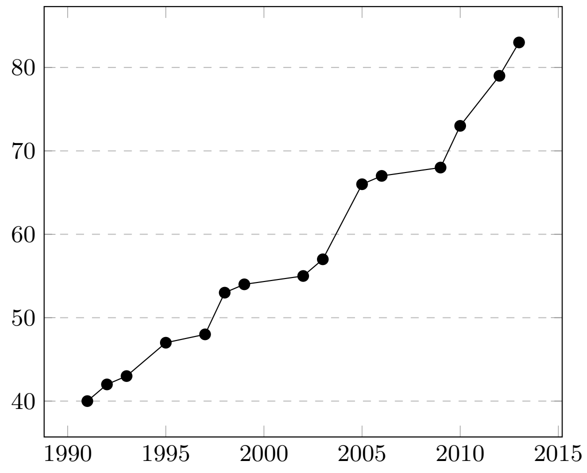


Fig. 2: Size of general number factored using ECM versus year

Therefore, the updated equation for the NFS is

$$D_n^{1/3} = \frac{Y - 1926}{13.97} \text{ or equivalently } Y = 13.97 \cdot D_n^{1/3} + 1926 \tag{6}$$

and for the ECM

$$D_p^{1/2} = \frac{Y - 1939}{8.207} \text{ or equivalently } Y = 8.207 \cdot D_p^{1/2} + 1939. \tag{7}$$

Equations (4) to (7) are presented in Figures 3 and 4. Note that the black dots represent the acquired data points. We observe that in the case of the NFS the estimates are close, while in the case of the ECM the new estimate is more pessimistic from a security point of view. Using the updated estimates (Equations (6) and (7)) we obtain the following equivalency

$$D_p^{1/2} = \frac{13.97 \cdot D_n^{1/3} - 13}{8.207}. \tag{8}$$

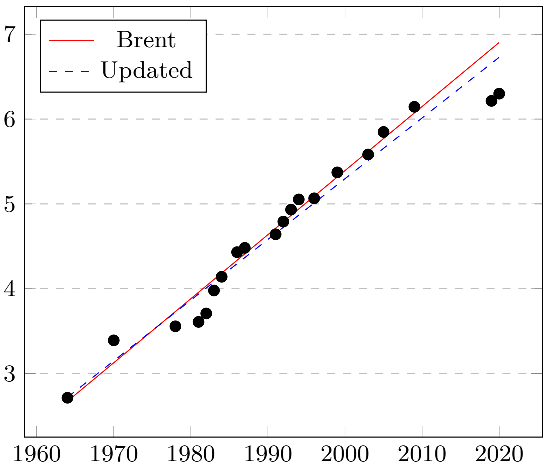


Fig. 3: $D^{1/3}$ versus year Y

According to NIST [6], the recommended key sizes for composite modules are $\lambda_n = 1536/3840/15360$. We preferred to use NIST recommendations instead of the ones from [25,26] since these key sizes are the ones used by the industry and the key sizes from [25,26] are criticized as being too conservative [32]. Therefore, using Equations (3) and (8) we obtain the equivalent size of the smallest prime. The results are presented in Table 2. Note that in the parentheses we provide the maximum number of prime factors that n can have. Based on these equivalences, we obtain the parameters for the Joye-Libert schemes that offer protection against the NFS and the ECM (see Table 4).

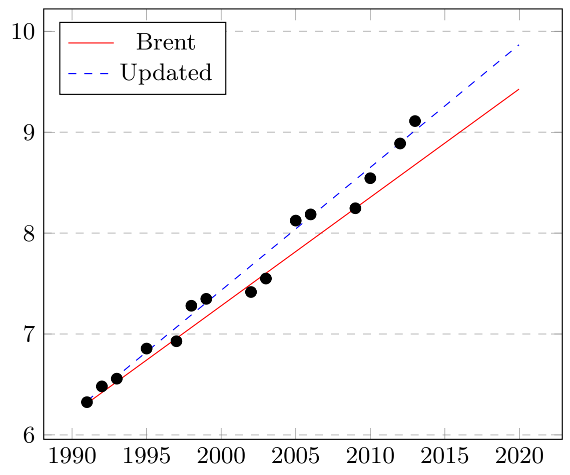


Fig. 4: $D^{1/2}$ versus year Y for ECM

Due to a powerful attack by Coppersmith [14], the size of k must be upper bounded by $0.5\lambda_p$. Otherwise, the factors of n can be found. We can easily see that the block sizes from Table 4 offer a large enough security margin obtained from this bound (see Table 3).

Modulus key size	3072	7680	15360
Lenstra model	800(3)	1617(4)	2761(5)
Regression model	749(4)	1457(5)	2385(6)

Table 2: Equivalent key sizes

$\|p\|$	1536	800	749	3840	1617	1457	7680	2761	2385
$0.5\|p\|$	768	400	374	1920	808	728	3840	1380	1192

Table 3: Coppersmith's upper bound

6.2 Complexity

Using the complexities provided in Table 1, we compute the asymptotic run times of the decryption algorithm for each Joye-Libert variant. We also determine the size of a block m_i for each variant. The results are provided in Table 5. Note

$\lvert n\rvert$	t	$\lvert p\rvert$	$\lvert q\rvert$	$\lvert m_i\rvert$	Type	Model
3072	1	1536	1536	128	Balanced	-
	1	800	2272	128	Unbalanced	Lenstra
	2	800	1472	64	Multiprime	
	1	749	2323	128	Unbalanced	Regression
	2	749	1574	64	Multiprime	
	3	749	825	43		
7680	1	3840	3840	192	Balanced	-
	1	1617	6063	192	Unbalanced	Lenstra
	2	1617	4446	96	Multiprime	
	3	1617	2829	64		
	1	1457	6223	192	Unbalanced	Regression
	2	1457	4766	96	Multiprime	
	3	1457	3309	64		
	4	1457	1852	48		
15360	1	7680	7680	256	Balanced	-
	1	2761	12599	256	Unbalanced	Lenstra
	2	2761	9838	128	Multiprime	
	3	2761	7077	86		
	4	2761	4316	64		
	1	2385	12975	256	Unbalanced	Regression
	2	2385	10590	128	Multiprime	
	3	2385	8205	86		
	4	2385	5820	64		
	5	2385	3435	52		

Table 4: Joye-Libert parameters' size

Scheme	$\lvert m\rvert$	Encryption Complexity	Decryption Complexity
Balanced	k	$\mathcal{O}(2kM(\lambda_n))$	$\mathcal{O}((2k\lambda+k)M(\lambda))$
			$\mathcal{O}((\lambda+k^2/2+3k/2)M(\lambda))$
Unbalanced	k	$\mathcal{O}(2kM(\lambda_n))$	$\mathcal{O}((2k\lambda_p+k)M(\lambda_p))$
			$\mathcal{O}((\lambda_p+k^2/2+3k/2)M(\lambda_p))$
Multiprime	tk	$\mathcal{O}(2ktM(\lambda_n))$	$\mathcal{O}(t(2k\lambda_p+k)M(\lambda_p))$
			$\mathcal{O}(t(\lambda_p+k^2/2+3k/2)M(\lambda_p))$
Parallel Multiprime	tk	$\mathcal{O}(2kM(\lambda_n))$	$\mathcal{O}((2k\lambda_p+k)M(\lambda_p))$
			$\mathcal{O}((\lambda_p+k^2/2+3k/2)M(\lambda_p))$

Table 5: Performance analysis

that by parallel multiprime we mean the multiprime version in which we use a separate thread to compute each block m_i. We can easily see that for the parameters presented in Table 4, the encryption and decryption complexities of the unbalanced and multiprime versions are similar. Therefore, we only compare the balanced, unbalanced and the parallel multiprime versions. Also, remark that we choose the parameters such that the message spaces are similar for all the variants.

The comparison of the computational complexity of the three variants is presented in Figures 5 to 7. Note that the two sets of crosses for the multiprime version correspond to the two equivalence models: Lenstra - right side crosses and Regression - left side crosses. We also added a dotted red line in the case of the basic decryption (Algorithm 1), which represent the boundary of the optimized decryption algorithm (Algorithm 2) of the balanced version.

From the six plots we can see that the parallel multiprime version always performs better than the (un)balanced version if multiple threads are available. Also, the more threads we use, the faster we recover the original message. Therefore, if additional memory is available and parallelization is possible, then the parallel multiprime version endowed with the optimized decryption algorithm is preferable. Nevertheless, if we only have access to parallelization, then the parallel multiprime version equipped with the basic decryption algorithm is the best choice. Otherwise, we should use the unbalanced variant.

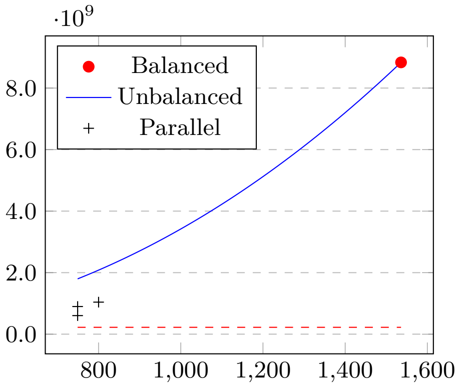
6.3 Implementation Details

We further provide the reader with benchmarks for the three Joye-Libert PKE schemes. We ran each of the three sub-algorithms on a CPU Intel i7-4790 4.00 GHz and used GCC to compile it (with the O3 flag activated for optimization). Note that for all computations we used the GMP library [2]. To calculate the running times we used the *omp_get_wtime()* function [1]. For the parallel multiprime variant we used the OMP library [1] to parallelize encryption/decryption. To obtain the average running time in seconds we chose to encrypt 100 128/192/256-bit messages. Therefore, we wanted to simulate a key distribution scenario. The results are provided in Table 6. Note that the optimized version of the decryption algorithm is denoted by *Decrypt* (opt).

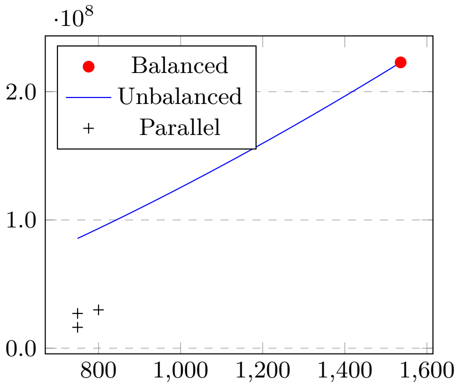
We can see from Table 6 that the conclusions presented in Section 6.2 hold. We can also see that the multiprime version has the shortest time to generate the parameters, while the unbalanced version the longest time. Nevertheless, generating parameters is a one-time operation.

7 Conclusions

In this work we introduced two novel versions of the Joye-Libert cryptosystem. The first one, called the unbalanced Joye-Libert PKE, lowers the size of p in order to decrease decryption time. The second one, called the multiprime Joye-Libert PKE, increases the number of factors and achieves better decryption

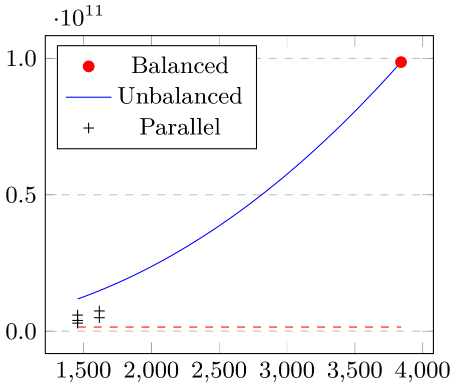


(a) Basic

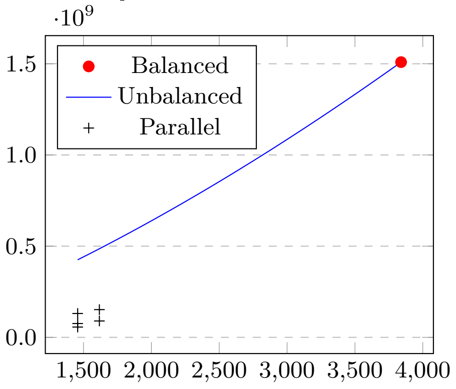


(b) Optimized

Fig. 5: Decryption complexity versus λ_p for $\lambda_n = 3072$

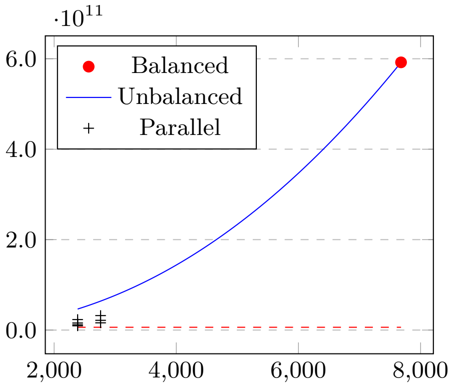


(a) Basic

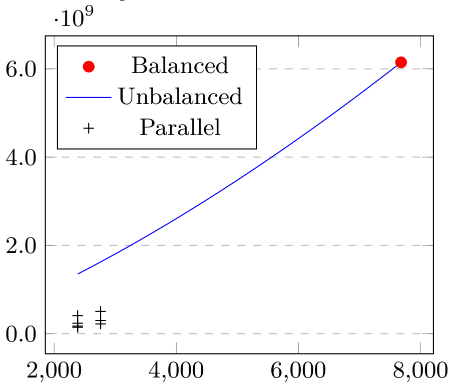


(b) Optimized

Fig. 6: Decryption complexity versus λ_p for $\lambda_n = 7680$



(a) Basic



(b) Optimized

Fig. 7: Decryption complexity versus λ_p for $\lambda_n = 15360$

$\|n\|$	t	$\|p\|$	*Setup*	*Encrypt*	*Decrypt*	*Decrypt* (opt)
3072	1	1536	0.193475	0.000782	0.325478	0.007954
	1	800	0.414516	0.000783	0.046704	0.002073
	2	800	0.129921	0.000584	0.025147	0.001427
	1	749	0.507081	0.000782	0.038254	0.001796
	2	749	0.140442	0.000567	0.020635	0.001273
	3	749	0.036575	0.000419	0.014810	0.000770
7680	1	3840	6.554910	0.004791	6.516350	0.095550
	1	1617	14.48600	0.004792	0.565853	0.017291
	2	1617	6.103730	0.003115	0.287015	0.006300
	3	1617	1.334260	0.002326	0.205834	0.004104
	1	1457	17.03100	0.004790	0.390024	0.013370
	2	1457	6.961240	0.003112	0.201755	0.004917
	3	1457	2.030210	0.002387	0.162581	0.003348
	4	1457	0.559723	0.001948	0.154818	0.002867
15360	1	7680	55.77040	0.018572	47.62750	0.472317
	1	2761	184.5970	0.018570	3.337000	0.080039
	2	2761	69.75550	0.010459	1.712070	0.029658
	3	2761	27.77600	0.007987	1.222490	0.023583
	4	2761	8.610660	0.006857	0.995357	0.018094
	1	2385	183.5240	0.018564	2.183190	0.059934
	2	2385	86.03020	0.010578	1.135120	0.021922
	3	2385	38.94160	0.008167	0.808199	0.017175
	4	2385	16.45910	0.006833	0.675422	0.012873
	5	2385	5.086330	0.006127	0.702819	0.012128

Table 6: Running times

times by using multiple threads. Therefore, if parallel threads are available, we recommend the multiprime version, otherwise, we recommend the unbalanced variant. If additional memory is available, then we can replace the basic decryption algorithm with the optimized version, and therefore we can get even better decryption times.

Open Problem. In [7], the authors manage to link the gap 2^k-residuosity assumption to the quadratic residuosity assumption, when $q \equiv 3 \bmod 4$ and to the quadratic residuosity and squared Jacobi symbols assumptions, when $q \equiv 1 \bmod 4$. Therefore, it would be interesting to find a similar link for the VQR assumption. The main bottleneck that we encountered when trying to link it to the VQR is that the probability of choosing an element from $J_n(i)$ is $1/2^{kt-k+2}$, and thus for t elements the probability is $1/2^{t(kt-k+2)}$. Therefore, for practical values of k and t, this probability is negligible.

References

1. OpenMP. https://www.openmp.org/
2. The GNU Multiple Precision Arithmetic Library. https://gmplib.org/
3. RSA Honor Roll. http://www.ontko.com/pub/rayo/primes/hr_rsa.txt (1999)
4. Cryptography Using Compaq Multiprime Technology in a Parallel Processing Environment. https://www.compaq.com (2000)
5. Bahr, F., Boehm, M., Franke, J., Kleinjung, T.: Factorization of RSA-200. https://members.loria.fr/PZimmermann/records/rsa200 (2005)
6. Barker, E.: NIST SP800-57 Recommendation for Key Management, Part 1: General. Tech. rep., NIST (2016)
7. Benhamouda, F., Herranz, J., Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. Journal of Cryptology **30**(2), 519–549 (2017)
8. Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., Zimmermann, P.: 795-bit Factoring and Discrete Logarithms. https://sympa.inria.fr/sympa/arc/cado-nfs/2019-12/msg00000.html (2019)
9. Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., Zimmermann, P.: Factorization of RSA-250. https://sympa.inria.fr/sympa/arc/cado-nfs/2020-02/msg00001.html (2020)
10. Brent, R.P.: Large Factors Found By ECM. https://maths-people.anu.edu.au/~brent/ftp/champs.txt
11. Brent, R.P.: Some Parallel Algorithms for Integer Factorisation. In: Euro-Par 1999. Lecture Notes in Computer Science, vol. 1685, pp. 1–22. Springer (1999)
12. Cavallar, S., Dodson, B., Lenstra, A., Leyland, P., Lioen, W., Montgomery, P.L., Murphy, B., Riele, H.t., Zimmermann, P.: Factorization of RSA-140 Using the Number Field Sieve. In: ASIACRYPT 1999. Lecture Notes in Computer Science, vol. 1716, pp. 195–207. Springer (1999)
13. Cavallar, S., Dodson, B., Lenstra, A.K., Lioen, W., Montgomery, P.L., Murphy, B., Riele, H.t., Aardal, K., Gilchrist, J., Guillerm, G., et al.: Factorization of a 512-bit rsa modulus. In: EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 1–18. Springer (2000)
14. Coppersmith, D.: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. Journal of Cryptology **10**(4), 233–260 (1997)
15. Crandall, R., Pomerance, C.: Prime Numbers: A Computational Perspective. Number Theory and Discrete Mathematics, Springer (2005)
16. Denny, T., Dodson, B., Lenstra, A.K., Manasse, M.S.: On the Factorization of RSA-120. In: CRYPTO 1993. Lecture Notes in Computer Science, vol. 773, pp. 166–174. Springer (1993)
17. Franke, J., Kleinjung, T., Montgomery, P., te Riele, H., Bahr, F., Leclair, D., Leyland, P., Wackerbarth, R.: Factorization of RSA-576. https://members.loria.fr/PZimmermann/records/rsa576 (2003)
18. Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In: STOC 1982. pp. 365–377. ACM (1982)
19. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)
20. Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. In: EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 76–92. Springer (2013)
21. Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. IACR Cryptology ePrint Archive **2013/435** (2014)

22. Jr., H.W.L.: Factoring Integers with Elliptic Curves. Annals of Mathematics pp. 649–673 (1987)
23. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., et al.: Factorization of a 768-Bit RSA Modulus. In: CRYPTO 2010. Lecture Notes in Computer Science, vol. 6223, pp. 333–350. Springer (2010)
24. Lenstra, A.K.: Unbelievable Security. Matching AES Security Using Public Key Systems. In: ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 67–86. Springer (2001)
25. Lenstra, A.K., Verheul, E.R.: Selecting Cryptographic Key Sizes. In: PKC 2000. Lecture Notes in Computer Science, vol. 1751, pp. 446–465. Springer (2000)
26. Lenstra, A.K., Verheul, E.R.: Selecting Cryptographic Key Sizes. Journal of Cryptology **14**(4), 255–293 (2001)
27. Maimuţ, D., Teşeleanu, G.: A New Generalisation of the Goldwasser-Micali Cryptosystem Based on the Gap 2^k-Residuosity Assumption. In: SecITC 2020. Lecture Notes in Computer Science, vol. 12596, pp. 24–40. Springer (2020)
28. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC press (1996)
29. Odlyzko, A.M.: The Future of Integer Factorization. RSA Laboratories' Cryptobytes **1**(2), 5–12 (1995)
30. Rivest, R.L., Shamir, A., Adleman, L.M.: Cryptographic Communications System and Method (1983), US Patent 4,405,829
31. Shamir, A.: RSA for Paranoids. RSA Laboratories' Cryptobytes **1**(3), 1–4 (1995)
32. Silverman, R.D.: A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths. RSA Laboratories' Bulletin 13 (2000)
33. Yan, S.Y.: Number Theory for Computing. Theoretical Computer Science, Springer (2002)
34. Zimmermann, P.: 50 Largest Factors Found by ECM. https://members.loria.fr/PZimmermann/records/top50.html