CryptoComputing with rationals

Pierre-Alain Fouque[1,2], Jacques Stern[2], and Geert-Jan Wackers[3]

[1] D.C.S.S.I. Crypto Lab
51, bd Latour-Maubourg, F-75007 Paris, France
[2] École Normale Supérieure, Département d'Informatique
45, rue d'Ulm, F-75230 Paris Cedex 05, France
{Pierre-Alain.Fouque,Jacques.Stern}@ens.fr
[3] Ernst & Young EDP Audit,
The Netherlands
{nlwackel}@ey.nl

Abstract. In this paper we describe a method to compute with encrypted *rational* numbers. It is well-known that *homomorphic schemes* allow calculations with hidden *integers*, *i.e.* given integers x and y encrypted in $\mathcal{E}(x)$ and $\mathcal{E}(y)$, one can compute the encrypted sum $\mathcal{E}(x+y)$ or the encrypted product $\mathcal{E}(kx)$ of the encrypted integer x and a known integer k without having to decrypt the terms $\mathcal{E}(x)$ or $\mathcal{E}(y)$. Such cryptosystems have a lot of applications in electronic voting schemes, lottery or in multiparty computation since they allow to keep the privacy of the terms and return the result in encrypted form. However, from a practical point of view, it might be interesting to compute with *rationals*. For instance, a lot of financial applications require algorithms to compute with rational values instead of integers such as bank accounts, electronic purses in order to make payments or micropayments, or secure spreadsheets. We present here a way to solve this problem using the Paillier cryptosystem which offers the largest bandwidth among all homomorphic schemes. The method uses two-dimensional lattices to recover the numerator and denominator of the rationals. Finally we implement this technique and our results in order to build an encrypted spreadsheet showing the practical possibilities of the homomorphic properties applied on rationals.

1 Introduction

A lot of financial applications use calculations such as sums of expenses, or multiplications by a public constant (rates). It is easy to show that electronic purses or bank accounts have to be encrypted from the users or clients point of view whereas some values can be public such as interest rates in order to update accounts at the end of each month. Therefore, we need to add or subtract two encrypted expenses (rational numbers) or to multiply encrypted numbers by a public rational. For example, if a bank wants to keep secret the accounts of its clients to the bank employees, a safe way consists in encrypting the bank accounts. There are two problems to solve : the first challenge is to encode rationals r such that the homomorphic properties of the cryptosystems still hold, and the second challenge is to reconstruct the numerator and denominator of the encrypted rational when recovery of the rationals is needed.

The basic idea to reduce the problem to compute with integers does not work. Indeed, one can think to multiply each value by 10^5 before encrypting them, provided we allow a precision of 10^{-5}. In this case, one can perform additions but *not* multiplications by a scalar. Consequently, one must use an encoding that respect both the multiplication and addition.

We can also remark that the encoding of the numerator and the denominator does not provide a valid solution. Indeed, let us consider the following encoding of the rational a/b as $[\mathcal{E}(a), \mathcal{E}(b)]$. If we have a multiplicative homomorphic scheme as the plain-RSA, then the multiplication of two rationals is a trivial task. However, the addition of two rationals is not possible, and furthermore, the addition is the more useful operation in spreadsheet.

1.1 Related Works

From a theoretical point of view, it is important to securely compute with encrypted data. Computing with hidden values or securely doing mathematical operations such as comparison of two encrypted numbers are deep operations in cryptography. The latter is known as the millionaire problem. It has been extensively studied to build efficient auction protocols. For example, at the last Financial Crypto, Baudron and Stern [3] have devised an algorithm to solve this problem. The problem is the following : given a set of encrypted bids $\{\mathcal{E}(v_i)\}_i$, find $\max(\mathcal{E}(v_i))$ without decrypting the bids. The important point of the Baudron and Stern algorithm is that the calculation is done without interactions between the participants.

The former problem of "computing with encrypted data" has been studied by numerous authors in the past twenty years. Sander, Young and Yung have proposed in [20] a protocol that non-interactively computes any function in $\mathcal{NC}^1$. This is an important result as it is an open problem to reduce the round of computation in general multiparty computation protocols. It is a particular case of the more general classical problem : Alice has an input x and Bob has circuit C. Alice should known the value $C(x)$ but nothing else "substantial" about C. Bob should learn nothing else "substantial" about Alice's input x. In our case, the function C can be learnt by anyone. Several papers more extensively discuss secure circuit evaluation such as [1].

Informally a cryptosystem is algebraically homomorphic if given the encryption of two plaintexts x and y, one can construct the encryption of the plaintexts $x + y$ and xy in polynomial time without revealing x or y. The existence of these schemes has been left open. In 1978, Rivest, Adleman, and Dertouzos [19] suggested investigating encryption schemes with additional homomorphic properties since they allow to compute with encrypted data. In 1991, Feigenbaum and Merritt [11] directly addressed algebraic homomorphic schemes of the form stated above. In [5], Boneh and Lipton showed that *deterministic* algebraically homomorphic encryption schemes over ring $\mathbb{Z}_N$ can be broken in subexponential time under a (reasonable) number theoretic assumption. In their argument, it is essential though, that the scheme be deterministic.

Homomorphic cryptosystems provide an efficient way to solve the "Secure Function Evaluation" Problem for some mathematical operations. In particular, we focus here on, $(\times, +)$-homomorphic schemes such as [4, 15–17], which are probabilistic and enable to perform addition of two encrypted values or multiplication by a known integer. Homomorphic schemes were proposed to solve more efficiently the problem of computing with integers under computational assumptions. These special kind of public-key cryptosystems have applications in electronic schemes or lotteries [4, 9, 12, 2]. These algorithms can also be used to build efficient multiparty computation protocols with reduced communication complexity [8].

Finally, Poupard and Stern used lattices and rationals to build a key recovery system in [18].

1.2 Our Results

In this paper, we study a new version of a cryptocomputer. Instead of searching cryptosystems which only allow to perform $+$ and $\times$ operations over the integers, we cover additions and multiplications by a scalar of bounded rational numbers. We can thus construct an encrypted spreadsheet with bounded rationals and analyze its implementation. A limitation of our paper is to use *bounded* rationals. In fact, we need to recover the encoded rationals using a lattice and the output of the Gauss algorithm depends on the size of the shortest vector. However, thanks to the large bandwidth of Paillier scheme, a large number of operations can be done.

1.3 Notations and Definitions

Throughout this paper, we use the following notation: for any integer N,

- we use $\mathbb{Z}_N$ to denote the set of the integers modulo N,
- we use $\mathbb{Z}_N^*$ to denote the multiplicative group of invertible elements of $\mathbb{Z}_N$,
- we use $\varphi(N)$ to denote the Euler totient function, i.e. the cardinality of $\mathbb{Z}_N^*$,
- we use $\lambda(N)$ to denote Carmichael's lambda function defined as the largest order of the elements of $\mathbb{Z}_N^*$.

It is well known that if the prime factorization of an odd integer N is $\prod_{i=1}^{\eta} {q_i}^{f_i}$ then $\varphi(N) = \prod_{i=1}^{\eta} {q_i}^{f_i-1}(q_i - 1)$ and $\lambda(N) = \mathrm{lcm}_{i=1\ldots\eta}\left({q_i}^{f_i-1}(q_i - 1)\right)$.

1.4 Outline of the paper

In section 2 we recall preliminary tools such as the Paillier cryptosystem, homomorphic properties and the Gauss algorithm. Then in section 3, we show how to encode and decode rationals bounded by integers modulo N. Next, we describe additions and products in section 4. Finally, we describe practical parameters and our spreadsheet application.

2 Preliminary tools

2.1 The Paillier cryptosystem

Various cryptosystems based on randomized encryption schemes $\mathcal{E}(M)$ which encrypt a message M by raising a basis g to the power M have been proposed so far [13, 4, 7, 23, 15–17]. Their security is based on the intractability of computing discrete logarithm in basis g without a secret element, the secret key. Given this secret as a trapdoor, the computation becomes easy. We call those cryptosystems *trapdoor discrete logarithm schemes*. As an important consequence of this encryption technique, those schemes have homomorphic properties that can be informally stated as follows:

$$\mathcal{E}(M_1 + M_2) = \mathcal{E}(M_1) \cdot \mathcal{E}(M_2) \quad \text{and} \quad \mathcal{E}(k \cdot M) = \mathcal{E}(M)^k$$

Paillier has presented three closely related such cryptosystems in [17]. We only recall the first one. This cryptosystem is based on the properties of the Carmichael lambda function in $\mathbb{Z}^*_{N^2}$. We state its main two properties: for any $w \in \mathbb{Z}^*_{N^2}$,

$$w^{\lambda(N)} = 1 \bmod N, \quad \text{and} \quad w^{N\lambda(N)} = 1 \bmod N^2$$

Key Generation Let N be an RSA modulus $N = pq$, where p and q are prime integers. Let g be an integer of order $N\alpha$ modulo N^2. The public key is $\mathsf{pk} = (N, g)$ and the secret key is $\mathsf{sk} = \lambda(N)$.

Encryption To encrypt a message $M \in \mathbb{Z}_N$, randomly choose x in $\mathbb{Z}^*_N$ and compute the ciphertext $c = g^M x^N \bmod N^2$.

Decryption To decrypt c, compute $M = \dfrac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \bmod N$ where the L-function takes as input an element from the set $\mathcal{S}_N = \{u < N^2 | u = 1 \bmod N\}$ and computes $L(u) = \frac{u-1}{N}$.

The integers $c^{\lambda(N)} \bmod N^2$ and $g^{\lambda(N)} \bmod N^2$ are equal to 1 when they are raised to the power N so they are N^{th} roots of unity. Furthermore, such roots are of the form $(1+N)^\beta = 1 + \beta N \bmod N^2$. Consequently, the L-function allows to compute such values $\beta \bmod N$ and $L((g^M)^{\lambda(N)} \bmod N^2) = M \cdot L(g^{\lambda(N)} \bmod N^2) \bmod N$.

One can note that g can be set to $1 + N$ in order to make encryption more efficient since $\mathcal{E}(m, x) = (1 + mN)x^N \bmod N^2$.

Security. It is conjectured that the so-called composite residuosity class problem, that exactly consists in inverting the cryptosystem, is intractable. This problem is easier than inverting RSA with public exponent N modulo N, but it is unknown whether the two problems are equivalent. The semantic security is based on the difficulty to distinguish N^{th} residues modulo N^2. We refer to [17] for details.

2.2 The Gauss algorithm

The Gauss algorithm solves the problem of finding a basis in a 2-dimensional lattice: it computes a basis achieving the two first minima. One can prove that the algorithm outputs such basis in polynomial time [14, 22]. Vallée in [22] has proved that the number of iterations is in the worst case $3+\log_{1+\sqrt{2}}\max(\|\mathbf{u}\|,\|\mathbf{v}\|)$, and the number of iterations is constant on average [10]. Finally, let (u',v') be the first vector of the reduced basis. It is the shortest vector in the lattice; therefore $\gcd(u',v')=1$.

Input: a basis $(\mathbf{u},\mathbf{v})$ of a lattice L
Output: a reduced lattice basis $(\mathbf{u},\mathbf{v})$
if $\|\mathbf{u}\|<\|\mathbf{v}\|$, then interchange $\mathbf{u}$ and $\mathbf{v}$

do
  $\mathbf{r}\leftarrow\mathbf{u}-q\mathbf{v}$ where $q=\left\lfloor\frac{\langle\mathbf{u},\mathbf{v}\rangle}{\|\mathbf{v}\|^2}\right\rceil$
  $\mathbf{u}\leftarrow\mathbf{v}$
  $\mathbf{v}\leftarrow\mathbf{r}$
until $\|\mathbf{u}\|\leq\|\mathbf{v}\|$
return $(\mathbf{u},\mathbf{v})$

Where $\lfloor x\rceil$ represents the integer closest to the real x.

3 Encoding and decoding bounded rationals by integers modulo N

In the Paillier cryptosystem presented above, the message M to be encrypted should be an integer modulo N. In this section we explain how a *bounded* rational t can be encrypted and recovered using the Paillier cryptosystem.

3.1 Encoding rationals in integers

Encoding of bounded rational. Let a bounded rational number $t=r/s$ where $r,s\in\mathbb{Z}$, $s\neq 0$, $-R<r<R$, $0<s<S$ $\gcd(r,s)=1$, and $\gcd(s,N)=1$ since $s<p$ and $s<q$. We denote by t' the representation of t in $\mathbb{Z}_N$ which is defined as :

$$t'=rs^{-1}\bmod N \tag{1}$$

The integer s^{-1} exists since $\gcd(s,N)=1$. By calculating $rs^{-1}\bmod N$, an integer $t'\bmod N$ is obtained which can be used for encrypting the rational t with the Paillier cryptosystem.

One can note that this encoding of rational respects the classical operations, namely if $\mathcal{E}(r)$ is a encrypted rational and $\mathcal{E}(i)$ an encrypted integer, then $\mathcal{E}(r)\times\mathcal{E}(i)=\mathcal{E}(r+i)$ and $\mathcal{E}(r)/\mathcal{E}(i)=\mathcal{E}(r-i)$.

Decoding of bounded rational. When retrieving the message information with the decrypting formula, the resulting message M equals $t' \bmod N$, *i.e.* $rs^{-1} \bmod N$.

Now the rational t must be recovered from $rs^{-1} \bmod N$. Recovery consists in fact in recovering $r < R$ and $s < S$ from $rs^{-1} \bmod N$. Recovery of r and s and thus of t can be correctly done by using two-dimensional lattice theory.

A lattice is a subgroup of some power $\mathbb{Z}$. It can be represented by a basis, two non-colinear vectors of the lattice.

Consider the two-dimensional lattice L defined as:

$$L : \forall (x, y) \in \mathbb{Z}^2, x = yt' \bmod N \tag{2}$$

The lattice L could then also be defined as:

$$L : \forall (x, y) \in \mathbb{Z}, sx = yr \bmod N \tag{3}$$

From formula 2 can be deduced that the vectors $(N, 0)$ and $(t', 1)$ form a basis of the two-dimensional lattice L.

From formula 3 can be deduced that the vector (r, s) also is a vector of the lattice L. Moreover, in order to obtain optimal values for r and s, the vector (r, s) should be a minimal vector of the lattice L. The shortest vector can be computed from the original basis and thus from t' and N by using the algorithm of Gauss [6]. The first vector of the basis corresponds to this rational. Therefore, we know that $\gcd(r, s) = 1$ and we obtain numerator and denominator that are coprime.

3.2 Decoding rationals in integers

In the above section we showed how lattice theory can be used to encode and recover rationals using the Paillier cryptosystem. In this section, we prove the unicity of the recovered solution (r, s) if the conditions of Theorem 1 are respected.

Theorem 1. *If $t' = rs^{-1} \bmod N$, $-R \leq r \leq R$ and $0 < s \leq S$, then the algorithm of Gauss uniquely recovers r and s provided $2RS < N$.*

Proof. We first prove that there cannot exist two linearly independent solutions in the lattice L generated by $(N, 0)$ and $(t', 1)$. Let (r_1, s_1) and (r_2, s_2) with $-R \leq r_i \leq R$ and $0 < s_i \leq S$ two different solutions. Such (r_1, s_1) and (r_2, s_2) would form a basis of a sub-lattice L' of the lattice L. The determinant N of the basis of L divides the determinant of the basis of L':

$$N \,|\det(L') \;\text{ where }\; \det(L') = \begin{vmatrix} r_1 & s_1 \\ r_2 & s_2 \end{vmatrix} = |r_1 s_2 - r_2 s_1|$$

Furthermore:

$$|r_1 s_2 - r_2 s_1| \leq 2RS < N$$

by assumption.

Therefore,

$$N \,|\det(L') < N \tag{4}$$

From formula 4, one can conclude that the determinant of the basis L' equals 0 and therefore the vectors (r_1, s_1) and (r_2, s_2) are colinear and cannot form a basis.

Thus, if the Gauss algorithm outputs (r, s) of the proper form, we are done. Otherwise, we have to argue in a more subtle manner and consider the lattice $\tilde{L}$ consisting of pairs (Sx, Ry) with $(x, y) \in L$. The determinant of this lattice is $\Delta = SRN$ and its shortest vector is of norm $\leq \sqrt{2}RS$ (coming from the encrypted rational). As before, it can be shown that there cannot be two linearly independent vectors of norm $\sqrt{2}RS$ since they would generate a sublattice with determinant $2R^2S^2 < \Delta$. Thus the Gauss algorithm run on $\tilde{L}$ outputs the correct rational number.

In order to recover the rational we need to restrict the rational to have numerator smaller than R and denominator smaller than S. Such numbers will be hereafter called *bounded rationals*.

4 Addition and multiplication with rationals

Applying Theorem 1, the implications for the homomorphic addition and product properties of the Paillier cryptosystem can be computed.

4.1 Addition

Lemma 1. *The algorithm of Gauss uniquely recovers the sum of ℓ terms of the form $r_i s_i^{-1}$ with $-R \leq r_i \leq R$ and $0 \leq s_i \leq S$ if $2(\ell+1)RS^{2\ell-1} < N$.*

Proof. Adding $r_1 s_1^{-1}$, $r_2 s_2^{-1}$, ..., and $r_\ell s_\ell^{-1}$ where $-R \leq r_i \leq R$ and $0 \leq s_i \leq S$ for $i = 1, \dots, \ell$, results in:

$$\frac{r_1}{s_1} + \frac{r_2}{s_2} + \ldots + \frac{r_\ell}{s_\ell} = \frac{r_1 \prod_{i=2,\dots,\ell} s_i + r_2 \prod_{i=1,\dots,\ell, i\neq 2} s_i + \ldots + r_\ell \prod_{i=1,\dots,\ell-1} s_i}{\prod_{i=1}^{\ell} s_i} \tag{5}$$

So with $|r_1 \prod_{i=2,\dots,\ell} s_i + r_2 \prod_{i=1,\dots,\ell,i\neq 2} s_i + \ldots + r_\ell \prod_{i=1,\dots,\ell-1} s_i| \leq \ell R S^{\ell-1}$ and $|\prod_{i=1}^{\ell} s_i| \leq S^\ell$, theorem 1 implies that:

$$2\ell R S^{2\ell-1} < N \tag{6}$$

4.2 Multiplication

When we need to multiply a bounded rational $t = r/s$ by a known bounded rational $t' = r'/s'$, we transform t' into an integer in $\mathbb{Z}_N$ by computing $k = r' \times (s'^{-1} \bmod N) \bmod N$. Then, we raise $\mathcal{E}(r \times (s^{-1} \bmod N))$ to the power k.

Lemma 2. *The algorithm of Gauss uniquely recovers the product of two factors of the form $t_i = r_i s_i^{-1}$, r_i and s_i are integers and $-R < r_i < R$ and $0 < s_i < S$, if $2R^2S^2 < N$. If ℓ multiplications are done, the bound is $2R^\ell S^\ell < N$.*

Proof. Since the numerator is $r_1 r_2 < R^2$ and the denominator $s_1 s_2 < S^2$, we need to have $2R^2S^2 < N$. It is easy to show thanks to theorem 1, when we multiply ℓ bounded rationals, that

$$2(RS)^\ell < N \tag{7}$$

5 Choice of parameters

Formula 6 and formula 7, respectively for the addition and product operations, show the limitations of the recovery method. Once the limit is reached, the homomorphic properties cannot be used anymore: a decryption and encryption step is needed before processing more operations.

For the addition can be concluded that the S parameter should be kept as small as possible in order to optimize the number of computations on ciphered data. For the product the numerator and denominator of the known rational should be kept as small as possible in order to optimize the number of computations on ciphered data. Indeed, before the numerator or denominator of the encrypted rational will be larger than the bound R or S, the spreadsheet must decrypt the rational, approximate it by a rational which have smaller numerator and denominator. We call this phase the "canonical form algorithm". One can use the continued fraction algorithm to approximate the decrypted rational by a bounded rational with smaller numerator and denominator. One can see that if we fix $|N| = 1024$, $R = 10^9 < 2^{30}$ and $S = 10^5 < 2^{17}$, one can only accept $\ell < \lfloor \frac{|N|-|R|-1}{2|S|} \rfloor + 1 \approx 30$ additions or $\ell < \lfloor \frac{|N|-1}{|R|+|S|} \rfloor \approx 21$ multiplications. This impacts the number of additions and multiplications that can be executed before the running of the canonical form algorithm.

A way to increase the number of possible additions or multiplications before runnig the canonical form algorithm is to use a cryptosystem with larger bandwidth. To this task, one can use a modulus with larger size or the technique of Damgård and Jurik [9] in order to increase the size of the bandwidth, N^{s-1}, using the same modulus N with computations done in N^s where s is some integer. This latter technique allows us to multiply the number of additions or multiplications by a factor $(s-1)$.

Finally, one can note that the bounds that we compute are not very tight. In practice, we can perform more additions or multiplications since the Gauss algorithm works nicer than the provable bounds.

6 Practical implementation

The method presented in the above sections has been successfully implemented in a Microsoft Excel 2000 spreadsheet running under Microsoft Windows 2000. This spreadsheet illustrates the properties as explained in the paper throug computations which encrypted data.

Excel does not support computations with large numbers as required by the Paillier cryptosystem. In order to solve this problem, the spreadsheet is only a graphical interface showing the process and the computation results. This is realized by implementing some self-designed buttons in the spreadsheet to which Visual Basic commands are associated. Those Visual Basic commands perform three different steps:

1. Data from specific spreadsheet cells is put in some ASCII files;
2. One or several executables are started in the Microsoft Windows 2000 environment;
3. The results of the executables which is also stored in ASCII files is retrieved and shown in the spreadsheet.

The executables in step 2 are compiled with Borland C++ 5.0 for Windows functions. In order to make the necessary computations with large numbers, the NTL library [21] of C++ functions is used. One of the biggest problems in the spreadsheet is to prevent Excel from starting step 3 before step 2 has finished as those steps are independent processes for the Windows Operating System. In order to prevent this and make the spreadsheet run smoothly, some timers have been added in the Visual Basic code.

Although the chosen method is not the most efficient and even looks a little cumbersome, it makes both parts of the implementation (the graphical interface and the computations) independent not only from each other but also from the operating system, thus facilitating their reuse in future projects.

The spreadsheet consists of four actual sheets:

1. The addition homomorphic property of the Paillier cryptosystem with integers;
2. The product homomorphic property of the Paillier cryptosystem with integers;
3. The addition homomorphic property of the Paillier cryptosystem with rationals and their recovery;
4. The limitation of the recovery method for the additional homomorphic property of the Paillier cryptosystem.

The spreadsheet confirms the results from the paper and illustrates the practical possibilities of the presented method to use the Paillier cryptosystem in order to compute with encrypted data consisting of rational numbers.

7 Conclusion

We proposed a method to use the Paillier cryptosystem and its homomorphic properties with rational numbers. The recovery of the numbers make use of two-dimensional lattice theory and especially the Gauss algorithm. Furthermore, we showed the conditions which have to be checked in order to uniquely recover the rationals uniquely. We implemented the method in a practical spreadsheet which makes computations on ciphered rationals and illustrates the limitations.

An open problem is to find a method to round or truncate the encrypted computation results, resetting the size of the parameters on the way.

References

1. M. Abadi and J. Feigenbaum. Secure circuit evaluation : a protocol based on hiding information from an oracle. *Journal of Cryptology*, 2(1):1–12, 1990.
2. O. Baudron, P.A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System. In *PODC '01*. ACM, 2001.
3. O. Baudron and J. Stern. Non-interactive Private Auctions. In *Financial Crypto '01*, LNCS. Springer-Verlag, Berlin, 2001.
4. J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, 1987.
5. D. Boneh and R. Lipton. Searching for Elements in Black-Box Fields and Applications. In *Crypto '96*, LNCS 1109, pages 283–297. Springer-Verlag, 1996.
6. H. Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics 138. Springer-Verlag, 1993.
7. J. Cohen and M. Fisher. A robust and verifiable cryptographically secure election scheme. In *Symposium on Foundations of Computer Science*. IEEE, 1985.
8. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Eurocrypt '01*, LNCS 2045, pages 280–300. Springer-Verlag, 2001.
9. I. Damgård and M. Jurik. Efficient Protocols based on Probabilistic Encryption using Composite Degree Residue Classes. In *PKC '01*, LNCS 1992, pages 119–136. Springer-Verlag, 2001.
10. H. Daudé, P; Flajolet, and B. Vallée. An average-case analysis of the gaussian algorithm for lattice reduction. *Combin. Probab. Comput.*, 6(4):397–433, 1997.
11. J. Feigenbaum and M. Merritt. Open Questions, Talks Abstracts, and Summary of Discussions. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2:1–45, 1991.
12. P. A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Crypto '00*, LNCS. Springer-Verlag, 2000.
13. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28, 1984.
14. A. Joux. *La Réduction des Réseaux en Cryptographie*. PhD thesis, École polytechnique, 1993.
15. D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proc. of the 5th CCCS*, pages 59–66. ACM press, 1998.
16. T. Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998.
17. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, 1999.
18. G. Poupard and J. Stern. Fair Encryption of RSA Keys. In *Proceedings of Eurocrypt 2000*, Lecture Notes in Computer Science, pages 172–189. Springer-Verlag, 2000.

19. R. Rivest, L. Adleman, and M. L. Dertouzos. On Data Banks and Privacy Homomorphisms. In *Foundations of Secure Computation*, pages 169–179. Academic Press, 1978.
20. T. Sander, A. Young, and M. Yung. Non-Interactive CryptoComputing for NC^1. In *Proc. of the 31st STOC*. ACM, 1999.
21. V. Shoup. Number Theory Library (NTL). Can be obtained at `http://www.shoup.net`.
22. B. Vallée. Gauss' algorithm revisited. *J. Algorithms*, 12:556–572, 1991.
23. S. Vanstone and R. Zuccherato. Elliptic Curve Cryptosystem Using Curves of Smooth Order Over the Ring Z_n. *IEEE Transaction on Information Theory*, IT-43, 1997.