Homomorphic Encryption Based on Group Algebras and Goldwasser-Micali Scheme

Cezar Pleşca[1,2], Mihai Togan[1,2](✉), and Cristian Lupaşcu[1]

[1] Computer Science Department, Military Technical Academy, Bucharest, Romania
cezar.plesca@gmail.com, mihai.togan@gmail.com, clupascu8@gmail.com
[2] certSIGN, Research and Development Department, Bucharest, Romania

Abstract. The possibility of outsourcing computation to the cloud offers businesses and individuals substantial cost-savings, flexibility, and availability of computable resources, but potentially sacrifices privacy. Homomorphic encryption can help address this problem by allowing the user to upload encrypted data to the cloud, on which the cloud can then operate without having the secret key. The cloud can return encrypted outputs of computations to the user without decrypting the data, thus providing data hosting and services without compromising privacy.

First, we present a general framework introduced in [3] which extends a group homomorphic encryption scheme with respect to one operation towards a cryptosystem having homomorphic properties on both operations (i.e. addition and multiplication). Second, we describe the main contribution of this paper by showing how this framework can be applied to a well known homomorphic encryption scheme, Goldwasser-Micali, analyzing the proposed cryptosystem security and its possible applications.

Keywords: Homomorphic encryption · Group algebra · Probabilistic public-key cryptography · Quadratic residuosity problem

1 Introduction

The idea of efficient and secure algorithms to encrypt messages *and compute* efficiently any algebraic functions on encrypted data goes back to Rivest et al. [1]. Since then, many attempts to produce such encryption schemes have been made. Lately much of interest is drawn to this domain mainly due to two factors:

1. First, the use of a large database implies in practice to retrieve just partial information, so the need of doing what is called nowadays cloud computing is imperative.
2. Secondly, a partial progress in this direction has obtained a break-through by Gentry's result [2] on bootstrapable encryption schemes.

In the last two decades the researchers in the area of encryption and coding have been more and more divided into two main categories: researchers whose main goal are theoretical results and researchers who try to find practical approaches of these results. Gentry's result is of a theoretical nature and one can only implement what is called leveled fully homomorphic encryption with a relatively small efficiency.

From a practical point of view, Barcău and Paşol suggested in [3] that the efficiency of a homomorphic encryption scheme cannot just be considered as a function of the security parameters for which one can prove polynomial asymptotic. For this reason, the authors advise that the theoretical schemes proposed in the literature should be accompanied by explicit algorithms and tested for practical efficiency and security with a present day computer technology.

We build upon the theoretical work presented in [3] and propose a general framework able to construct a cryptosystem with homomorphic operations (i.e. addition and multiplication) based on a group homomorphic encryption scheme. Then, we apply this framework to a well known group homomorphic encryption scheme, namely Goldwasser-Micali [4], to produce and successfully implement a practical ring homomorphic encryption scheme.

1.1 State of the Art

One of the first algorithms which has the feature to perform algebraic computations on the encrypted data without revealing the encrypted information was proposed by Fellows and Koblitz in [5]. However, few years later, the algorithm proved to be insecure and no modifications of the algorithm could solve this inconvenient. In 1998, Hoffstein et al. [6] proposed a secure and efficient algorithm to encode messages called NTRU. It does have the same ring homomorphic feature, but it allows only a few operations (i.e. additions and multiplications) to be performed on the encrypted data. This *leveled* feature comes from the fact that the algorithm is an error-based one, so only circuits which keep the noise very low can be applied to the encrypted data.

A better use of the error-based encryption technique for the purpose of achieving fully homomorphic encryption scheme was proposed by Gentry in his Ph.D. thesis [2] where he used ideal lattices and latter in his work (together with his collaborators), Regev's learning with error theory to produce algorithms which accommodate a much larger number of computations on the encrypted data. He also proved that, if an algorithm has the capability of computing the polynomial corresponding to the extended decryption algorithm (i.e. bootstrapable encryption scheme), then one can use it in a limiting process to produce a fully homomorphic encryption scheme. The word limiting is important from the practical implementation point of view because, in this sense, one can achieve only leveled fully homomorphic encryption scheme, which means that one has to prescribe from the beginning, the degree (or the depth) of the polynomials to be computed on the encrypted data.

Since the Gentry's break-through, many improvements of the algorithms based on learning with errors theory have been published and a research team from IBM conducted by Halevi and Shoup proposed an implementation based on ideas found in [7–9]. The implementation, written in C++ and using the NTL library, is called Homomorphic-Encryption Library (HELib) [10].

Recently, the encryption community raised the question concerning the realization of at least a leveled fully homomorphic encryption scheme using algorithms that are not error-based. The error-based encryption algorithms have two major deficiencies: first, in order to accommodate the error, the fresh ciphertexts have to be quite large and secondly, by its nature, the algorithms produce only leveled encryption schemes and one needs the process of bootstrapping in order to accommodate the desired depth for computations, a process which proved to be extremely high resource-consuming.

In an attempt to answer this question, Barcău and Paşol [3] reused some ideas from Grigoriev and Ponomarenko's work [11] to propose a fully homomorphic encryption scheme using monoid or group algebras.

The basic idea is that if one has already an encryption scheme which supports an encrypted operation (and there exist many such encryption schemes in the literature), then, one can use the group algebra theory to obtain an encryption scheme which supports algebraic operations on the encrypted data. However, the algorithms described in [11] are not efficient and cannot be used to produce fully homomorphic encryption schemes. The blueprint in [3] is more general and flexible enough to overcome some of these drawbacks, proposing a general framework to produce fully homomorphic encryption schemes. We give in this paper one example based on Goldwasser-Micali cryptosystem [4], and analyze its security and efficiency properties.

The rest of this paper is organized as follows: first, we present the main definitions and mathematicals results about quadratic residues in Sect. 2, which are later used in the description of Goldwasser-Micali cryptosystem in Sect. 3. Then, Sect. 4 introduces the reader into the field of group algebras upon which our general framework for ring homomorphic schemes, described in Sect. 5, is built. The application of this framework to Goldwasser-Micali cryptosystem is presented in Sect. 6 and the analysis of its properties from a practical point of view is done in Sect. 7. Conclusions about the proposed general framework for ring homomorphic encryption together with some other directions for its practical application ends our paper.

2 Quadratic Residues, Legendre and Jacobi Symbols

Let $m, n \in \mathbb{Z}$ with $(m, n) = 1$. Then m is called a quadratic residue mod n if and only if $\exists x \in \mathbb{Z}$ such that $m \equiv x^2 \pmod{n}$; otherwise, m is called a quadratic non residue mod n. For odd prime p, it is easy to see that exactly half of the non-null residues mod p from $\mathbb{Z}_p^*$ are quadratic and the other half are not.

2.1 Legendre Symbol and Its Properties

For an odd prime p and $n \in \mathbb{Z}$, the **Legendre symbol** $(\frac{n}{p})$ is defined as:

$$\left(\frac{n}{p}\right) = \begin{cases} 1 & \text{if } n \text{ is a quadratic residue mod } p \\ -1 & \text{if } n \text{ is a quadratic non residue mod } p \\ 0 & \text{if } p|n. \end{cases}$$

Hereafter, we recapitulate some important properties of the Legendre symbol. Let p be an odd prime and let $m, n \in \mathbb{Z}$. Then the following are true:

$$\left(\frac{mn}{p}\right) = \left(\frac{m}{p}\right)\left(\frac{n}{p}\right) \tag{1}$$

$$m \equiv n \pmod{p} \Rightarrow \left(\frac{m}{p}\right) = \left(\frac{n}{p}\right) \tag{2}$$

Let p be an odd prime and let $n \in \mathbb{Z}$ with $(n, p) = 1$. Starting from Fermat's Little Theorem: $n^{p-1} \equiv 1 \pmod{p}$, one can deduce that the all $p-1$ non-null residues mod p are solutions of the equation $x^{p-1} = 1$ in $\mathbb{Z}_p$. One can use the factorization $x^{p-1} - 1 = (x^{(p-1)/2} - 1)(x^{(p-1)/2} + 1)$, and easily observe that quadratic residues from $\mathbb{Z}_p^*$ are roots of the polynomial $x^{(p-1)/2} - 1$.

Since the polynomial $x^{(p-1)/2} - 1$ can only have $(p-1)/2$ distinct roots in $\mathbb{Z}_p$, it remains that all other $(p-1)/2$ non quadratic residues from $\mathbb{Z}_p^*$ are roots for the other polynomial, namely $x^{(p-1)/2} + 1$. This leads us to the fundamental result about Legendre symbols, the Euler's Criterion: for an odd prime p and $n \in \mathbb{Z}$ with $(n, p) = 1$, we have:

$$\left(\frac{n}{p}\right) \equiv n^{\frac{p-1}{2}} \pmod{p}. \tag{3}$$

Plugging in various values for n into Eq. 3, one can get the following immediate consequences for an odd prime p:

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}, \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} \tag{4}$$

The law of quadratic reciprocity gives a relationship between the two Legendre symbols $(\frac{p}{q})$ and $(\frac{q}{p})$ for two distinct odd primes p and q [12]:

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}} \tag{5}$$

2.2 Jacobi Symbol and Its Properties

The Jacobi symbol is a generalization of the Legendre symbol, defined in the previous subsection. Let $n > 1$ be an odd integer with prime factorization $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$. Then, for any integer a, the Jacobi symbol is defined as:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

The Jacobi symbol $(\frac{a}{1})$ is defined to be 1 for any integer a. As a consequence, the Jacobi symbol $(\frac{a}{n}) \in \{0, +1, -1\}$ and for an integer $n > 1$, we have:

$$\left(\frac{a}{n}\right) = \begin{cases} 0 & \text{if } \gcd(a, n) \neq 1 \\ \pm 1 & \text{if } \gcd(a, n) = 1 \end{cases}$$

It is easy to show the following properties of the Jacobi symbol. Let m, n be any positive odd integers and a, b be any integers. Then we have:

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right), \left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{b}{n}\right), \left(\frac{a}{n}\right) = \left(\frac{a \bmod n}{n}\right) \tag{6}$$

$$\left(\frac{-1}{n}\right) = (-1)^{\frac{p-1}{2}}, \left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}} \tag{7}$$

$$\left(\frac{m}{n}\right) = (-1)^{\frac{n-1}{2} \cdot \frac{m-1}{2}} \left(\frac{n}{m}\right), \left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{n-1}{2} \cdot \frac{m-1}{2}} \tag{8}$$

The first three properties follow directly from the definition. Properties 7 and 8 could be deduced, by observing that, when all the primes p_i are odd, we have:

$$\left(\sum_{i=1}^{k} \frac{p_i^{e_i} - 1}{2} \mod 2\right) = \left(\frac{p_1^{e_1} p_2^{e_2} \ldots\ldots p_k^{e_k} - 1}{2} \mod 2\right)$$

2.3 Computing Jacobi Symbol

The Jacobi symbol $(\frac{a}{n})$ is easy to compute when the prime factorization of n is known. We now show how to compute it efficiently when this factorization is not known. This algorithm finds its importance in the next chapter where we'll describe the Goldwasser-Micali cryptosystem on which our proposal is based.

Let $n > 1$ be an odd integer and $a \in \mathbb{Z}_{+}^{*}$. Then we can write $a = 2^e \cdot n'$ with n' odd and $e \geq 0$, and we can write $n = qn' + a'$, with $0 \leq a' \leq n' - 1$. Then, from the properties of the Jacobi symbol described above, we obtain:

$$\begin{aligned} \left(\frac{a}{n}\right) = \left(\frac{2^e \cdot n'}{n}\right) = \left(\frac{2^e}{n}\right)\left(\frac{n'}{n}\right) = \left(\frac{2}{n}\right)^e \left(\frac{n'}{n}\right) \\ = (-1)^{\frac{e\left(n^2-1\right)}{8}} \left(\frac{n'}{n}\right) = (-1)^{e\frac{n^2-1}{8} + \frac{n-1}{2}\frac{n'-1}{2}} \left(\frac{n}{n'}\right) \\ = (-1)^{e\frac{n^2-1}{8} + \frac{n-1}{2}\frac{n'-1}{2}} \left(\frac{qn' + a'}{n'}\right) = (-1)^{e\frac{n^2-1}{8} + \frac{n-1}{2}\frac{n'-1}{2}} \left(\frac{a'}{n'}\right) \end{aligned}$$

The important thing is that the value a' is strictly smaller than $|a|$. If we continue this process, we will ultimately obtain $a' = 0$, in which case the Jacobi symbol is trivial to evaluate. Let us define the following function:

$$f(e, n, n') = (-1)^{e\frac{n^2-1}{8} + \frac{n-1}{2} \cdot \frac{n'-1}{2}}$$

It is easy to show that the value $f(e, n, n')$ depends only on e mod 2, n mod 8 and n' mod 4. We thus have the two following rules that enable us to compute the Jacobi symbol:

$$\left(\frac{a}{n}\right) = f(e, n, n') \left(\frac{a'}{n'}\right)$$

$$\left(\frac{0}{n'}\right) = \begin{cases} 1 & \text{if } n' = 1 \\ 0 & \text{if } n' \neq 1 \end{cases}$$

A careful analysis very similar to the analysis done for Euclid's algorithm for computing the greatest common divisor, actually shows that the running time of the procedure suggested above is $O((\log a)(\log n))$. The Jacobi symbol $(\frac{a}{n})$ can then be computed in time $O((\log a)(\log n))$.

3 Goldwasser-Micali Cryptosystem

The Goldwasser-Micali (GM) cryptosystem is an asymmetric key encryption algorithm developed by Shafi Goldwasser and Silvio Micali in 1982. GM has the distinction of being the first probabilistic public-key encryption scheme which is provably secure under standard cryptographic assumptions. However, it is not an efficient cryptosystem, as ciphertexts may be several hundred times larger than the initial plaintext. To prove the security properties of the cryptosystem, Goldwasser and Micali proposed the widely used definition of semantic security.

The GM cryptosystem is semantically secure based on the assumed intractability of the quadratic residuosity problem modulo a composite $N = pq$, where p and q are large primes. This assumption states that given the couple (x, N) it is difficult to determine whether x is a quadratic residue modulo N (i.e., $x = y^2$ mod N for some y), when the Jacobi symbol for x is $+1$.

The quadratic residue problem is easily solved given the factorization of N, while new quadratic residues may be generated by any party, even without knowledge of this factorization. The GM cryptosystem leverages this asymmetry by encrypting individual plaintext bits as either random quadratic residues or non-residues modulo N, all with quadratic residue symbol $+1$. Recipients use the factorization of N as a secret key, and decrypt the message by testing the quadratic residuosity of the received ciphertext values.

Because Goldwasser-Micali produces a value of size approximately $|N|$ to encrypt every single bit of a plaintext, GM encryption results in substantial ciphertext expansion. To prevent factorization attacks, it is recommended that $|N|$ be several hundred bits or more. Because encryption is performed using a probabilistic algorithm, a given plaintext may produce very different ciphertexts, thus offering significant advantages, as it prevents an adversary from recognizing intercepted messages by comparing them to a dictionary of known ciphertexts.

GoldwasserMicali consists of 3 algorithms: a probabilistic key generation algorithm which produces a public and a private key, a probabilistic encryption algorithm, and a deterministic decryption algorithm. The scheme relies on deciding whether a given value x is a square mod N, given the factorization $N = pq$.

3.1 Key Generation

The modulus used in GoldwasserMicali encryption scheme is generated in the same manner as in the RSA cryptosystem.

1. Alice generates two distinct large prime numbers p and q, randomly and independently of each other, then computes $N = pq$.
2. She then finds some non-residue x such that the Legendre symbols satisfy $(x/p) = (x/q) = -1$ and hence the Jacobi symbol (x/N) is $+1$.

The value x can for example be found by selecting random values and testing the two Legendre symbols. If $p, q \equiv 3 \bmod 4$ (i.e., N is a Blum integer), then the value $N - 1$ is guaranteed to have the required property. The public key consists of (x, N), while the secret key is the factorization (p, q).

3.2 Message Encryption and Decryption

Encryption. Suppose Bob wishes to send a message m to Alice. Bob first encodes m as a string of bits $(m_1, ..., m_n)$. For every bit m_i, Bob generates a random value y_i from the group of units modulo N, or $\gcd(y_i, N) = 1$. He outputs the value $c_i = y_i^2 x^{m_i} \pmod N$. Bob sends the ciphertext $c = (c_1, ..., c_n)$.

Decryption. Alice receives $(c_1, ..., c_n)$. She can recover m using the following procedure: for each i, using the prime factorization (p, q), Alice determines whether the value c_i is a quadratic residue; if so, $m_i = 0$, otherwise $m_i = 1$. Alice outputs the message $m = (m_1, ..., m_n)$.

4 Group Algebras

In this section we describe how one can associate to any finite group $(G, \cdot)$ and any abelian ring $(R, +, \cdot)$ a group algebra noted as $R[G]$. Further, we will explore its properties and show that $R[G]$ has a commutative ring structure.

4.1 Notations and Properties

Every element $r \in R[G]$ has a unique representation given by:

$$r = \sum_{g \in G} r_g [g] \tag{9}$$

where $[g](g \in G)$ stands for a symbolic element from G and r_g are coefficients from the ring R. Otherwise, one can interpret such an element r as a vector from R^G whose coordinates r_g are indexed by an order defined over the entire set G. Over $R[G]$, one can define the addition operation which corresponds to the vector component-wise addition from R^G, as follows:

$$a = \sum_{g \in G} a_g [g], b = \sum_{g \in G} b_g [g] \quad a + b = \sum_{g \in G} (a_g + b_g)[g] \tag{10}$$

The multiplication over $R[G]$ is defined by the R-bilinear extension of $[x] \cdot [y] = [xy]$, thus the product of $a, b \in R[G]$ is given by:

$$a = \sum_{g \in G} a_g[g], b = \sum_{h \in G} b_h[h]$$

$$ab = \sum_{g,h \in G} a_g b_h [gh] = \sum_{f \in G} \left(\sum_{gh=f} a_g b_h \right) [f] \tag{11}$$

Enriched with the two operations previously defined, one can easily verify that $R[G]$ has a ring structure, having the following important identity elements:

$$\begin{aligned} &\mathit{Addition}: && 0 = \sum_{g \in G} 0[g] \\ &\mathit{Multiplication}: && 1[e] = 1[e] + \sum_{g \in G \setminus \{e\}} 0[g] \end{aligned} \tag{12}$$

In the previous formula, e is the identity element of G, 0 and 1 are the identity elements of R, with respect to addition and multiplication respectively. One can notice that the $R[G]$ algebra is commutative if and only if G is commutative.

4.2 Homomorphism Between Group Algebras

Let's consider two abelian groups, $(G, \cdot)$ and $(H, *)$ with a group homomorphism $\phi : G \to H$. Let's consider also a commutative ring R and the two algebras, $R[G]$ and $R[H]$, defined as above. Then ϕ induces an R-algebra homomorphism via the application $\phi^R : R[G] \to R[H]$, defined as follows:

$$\phi^R \left(\sum_{g \in G} r_g[g] \right) = \sum_{g \in G} r_g[\phi(g)] = \sum_{h \in H} \left(\sum_{\phi(g)=h} r_g \right) [h] \tag{13}$$

Notice that formula (13) defines ϕ^R as the R-linear extension of ϕ. The homomorphic property of ϕ^R with respect to addition operation is proven as follows:

$$\begin{aligned} \phi^R \left(\sum_{g \in G} a_g[g] + \sum_{g \in G} b_g[g] \right) &= \phi^R \left(\sum_{g \in G} (a_g + b_g)[g] \right) = \sum_{g \in G} (a_g + b_g)[\phi(g)] \\ = \sum_{g \in G} a_g[\phi(g)] + \sum_{g \in G} b_g[\phi(g)] &= \phi^R \left(\sum_{g \in G} a_g[g] \right) + \phi^R \left(\sum_{g \in G} a_g[g] \right) \end{aligned} \tag{14}$$

The homomorphic property of ϕ^R with respect to multiplication operation is proven by the following two equations:

$$\begin{aligned} \phi^R \left(\sum_{g \in G} a_g[g] \cdot \sum_{h \in G} b_h[h] \right) &= \phi^R \left(\sum_{f \in G} \left(\sum_{gh=f} a_g b_h \right) [f] \right) \\ = \sum_{f \in G} \left(\sum_{gh=f} a_g b_h \right) [\phi(f)] &= \sum_{\phi(gh)} \left(\sum_{\phi(gh)=ct.} a_g b_h \right) [\phi(gh)] \end{aligned} \tag{15}$$

$$\phi^R\left(\sum_{g\in G} a_g[g]\right)\cdot\phi^R\left(\sum_{h\in G} b_h[h]\right)=\sum_{g\in G} a_g[\phi(g)]\cdot\sum_{h\in G} b_h[\phi(h)]$$
$$=\sum_{\phi(g)*\phi(h)}\left(\sum_{\phi(gh)=ct.} a_g b_h\right)[\phi(g)*\phi(h)]=\sum_{\phi(gh)}\left(\sum_{\phi(gh)=ct.} a_g b_h\right)[\phi(gh)] \quad (16)$$

Moreover, for any commutative ring R, there exists an evaluation map from $R[R]$ to R, that is the natural R-algebra homomorphism $\epsilon : R[R] \to R$ given by:

$$\epsilon\left(\sum_{x\in R} r_x[x]\right)=\sum_{x\in R} r_x x. \quad (17)$$

5 Ring Homomorphic Encryption Schemes

In this section we describe the ring homomorphic encryption schemes proposal presented in [3]. Let (G, H, E, D) be a group homomorphic encryption scheme, a commutative ring, R and a morphism $\chi : H \to (R, \cdot)$. We proceed by describing a ring encryption scheme $(R[G], R, \mathrm{Enc}, \mathrm{Dec})$, which we'll prove to be homomorphic in the sense that $\mathrm{Dec} : R[G] \to R$ is a ring homomorphism.

Consider the image S of H in R through χ, and consider a fixed tuple $(r_1, \ldots, r_k) \in R^k$, where $k \geq 2$, such that the set containing elements of the form $\sum_{i=1}^{k} r_i s_i$ with $s_i \in S$ (not necessarily distinct) is the whole ring R. We'll explain later how and why this coverage property is important for the security scheme. It is important to note that the homomorphic application $\chi : H \to (R, \cdot)$ should not be trivial and allows us to find a fixed tuple $(r_1, \ldots, r_k) \in R^k$.

Encryption. The encryption algorithm is described by the following steps:

1. For a plaintext $m \in R$ consider one tuple $(s_1, \ldots, s_k) \in S^k$ such that:

$$m=\sum_{i=1}^{k} r_i s_i \quad (18)$$

2. Choose $(h_1, \ldots, h_k) \in H^k$ such that $\chi(h_i) = s_i, \forall i \in \{1, \ldots, k\}$
3. The encryption of the plaintext $m \in R$ is the following expression from $R[G]$:

$$Enc(m) := \sum_{i=1}^{k} r_i[E(h_i)] \quad (19)$$

Decryption. The decryption of an element from $R[G]$ is defined by the formula:

$$Dec\left(\sum_{g\in G} r_g[g]\right) := \sum_{g\in G} r_g \chi(D(g)). \quad (20)$$

5.1 Homomorphic Properties of the Encryption Scheme

As we have seen in Sect. 4, given the homomorphic properties of the χ mapping and the decryption function D (for the initial scheme), we'll get that $Dec : R[G] \to R$ is actually a ring homomorphism. More specifically, considering two cipher-texts a and b from $R[G]$, we have the following property:

$$\begin{aligned} Dec(a+b) &= Dec\left(\sum_{g\in G} a_g[g] + \sum_{g\in G} b_g[g]\right) = Dec\left(\sum_{g\in G}(a_g+b_g)[g]\right) \\ &= \sum_{g\in G}(a_g+b_g)[\chi(D(g))] = \sum_{g\in G} a_g[\chi(D(g))] + \sum_{g\in G} b_g[\chi(D(g))] \\ &= Dec\left(\sum_{g\in G} a_g[g]\right) + Dec\left(\sum_{g\in G} b_g[g]\right) = Dec(a) + Dec(b) \end{aligned} \quad (21)$$

The homomorphic property of the decryption function with respect to multiplication is done in a similar manner as shown previously by Eqs. 15 and 16. The security of the scheme is the same as the security of the group encryption scheme (G, H, E, D) since no information and no additional security was revealed or added through the steps describing the encryption algorithm.

5.2 Security Considerations

The choice to generate the set $(r_1, \ldots, r_k)$ as it was described earlier ensures the privacy of the encryption scheme in the sense that any plaintext has the same probability of being encrypted. An attacker having the cipher-text encoded as a vector $[r_i, g_i = E(h_i)], 1 \le i \le k$, could attempt to evaluate the plaintext as a linear combination of r_i elements of the form $\sum_{i=1}^{k} r_i s_i$, $s_i \in S$.

As guaranteed by the initial group encryption scheme security, the attacker could not know anything regarding h_i, henceforth he or she knows nothing about the $s_i \in S$. If all possible linear combinations of the form $\sum_{i=1}^{k} r_i s_i$ with $s_i \in S$ would not cover the entire set R, then at least the attacker could guess some information about the plaintext, i.e. knowing that certain plaintexts are for sure not encrypted in the given ciphertext. This is why we require the coverage property of R from the linear combinations of the form $\sum_{i=1}^{k} r_i s_i$ with $s_i \in S$.

A very important observation needs to be done: one should make the difference between the probability of plaintexts generated by choosing random elements in S and producing the plaintext $\sum_{i=1}^{k} r_i s_i$ and the probability of a certain plaintext to be encrypted. In essence, the choice of the set $(r_1, \ldots, r_k)$ ensures that no plaintext is left outside the encryption process.

Basically, the output of the encryption algorithm is a vector of GM-encryptions. Since the GM algorithm is a public-key encryption scheme, in order to decrypt the ciphertext, one has to decrypt each component of the vector. In other words, the security of the scheme is equivalent to the security of the GM-scheme.

5.3 Efficiency Considerations

In some cases, the choice of the of the generating set $(r_1, \ldots, r_k)$ could lead to a unique and deterministic choice for the $(s_1, \ldots, s_k)$, $s_i \in S$. Having such an efficient algorithm to find the unique linear combination of a plaintext obviously speeds up the encryption process. The bigger the set S is inside R the smaller the number k can be chosen (but not less than 2 if $S \subseteq R^*$ since for $k = 1$ the privacy will be breached by the fact that, in this case, the number 0 in the plaintext cannot be encrypted by a nonzero element in $R[G]$).

The parameter k has an impact over the length of ciphertexts, henceforth over the scheme efficiency. The efficiency of the encryption scheme is k times less the efficiency of the group homomorphic encryption scheme since basically the length of the ciphertext obtained by Enc is approximately $k\times$(the length of a ciphertext obtained by E plus the length of the message) (by coding the couples $\{r_i, g_i\}$). The decryption algorithm Dec has the speed of the algorithm D in the group homomorphic encryption scheme divided by the ciphertext dimension, i.e. the number of couples $\{r_i, g_i\}$ from the ciphertext.

Having fixed the encryption scheme, the length of the ciphertexts obtained by performing algebraic computations is *finite* since all computations take place in $R[G]$ which is a finite ring. An addition operation will lead very often to a cipher text whose length is the sum of the operands' lengths, since for multiplication, the length will grow up to the product of the ciphertexts' lengths.

One has to be caution in implementing the above scheme in cloud computing for the following reason: even though the ciphertext resulted by computing a polynomial on ciphertexts remains finite, its length is growing up to a certain point exponentially. The maximal length of an element in $R[G]$ is often huge for practical purposes. Therefore, for implementation, one would need an additional process, called *sparsification* in which one has to ensure a practical finiteness of an output after an algebraic manipulation on ciphertexts. To conclude, all of the algebraic properties as well as the properties required in the privacy, efficiency and security problems are satisfied by the ring homomorphic encryption scheme constructed above if one starts with an efficient, private and secure group homomorphic encryption scheme.

6 Homomorphic Encryption Using GM Scheme

As we have seen in Sect. 5, given the homomorphic properties of the decryption function $D : G \to H$ of some scheme, one can build a ring homomorphic encryption system by means of a homomorphic mapping $\chi : H \to (R, \cdot)$, where R is the ring corresponding to the plaintext space.

It is worth to mention that many of the encryption schemes already treated in the literature are in fact group homomorphic schemes: RSA, ElGamal, Paillier, Goldwasser-Micali, Benaloh, Diffie-Hellman, etc. Practical encryption schemes require additional constraints on the algorithms KeyGen, Enc and Dec such that the encryption and decryption processes are both feasible, secure and efficient.

We will show how the homomorphic encryption scheme Goldwasser - Micali could be extended to a ring homomorphic encryption using the calculus already presented in Sect. 5. More exactly, for an odd prime m, we consider the ring $R = \mathbb{Z}_m$, with both addition and multiplication done modulo m.

Now we consider the group homomorphic encryption scheme (G, H, E, D) described in Sect. 3 specific to the Goldwasser-Micali scheme where $G = (\mathbb{Z}_N, \cdot)$ and $H = (\{-1, 1\}, \cdot)$. The value of N is chosen as the product of two distinct large prime numbers p and q. Then, one finds some non-residue x such that $(\frac{x}{p}) = (\frac{x}{q}) = -1$ and hence the Jacobi symbol $(\frac{x}{N})$ is $+1$. The encryption and decryption function for the lightly modified GM scheme, E and D respectively, are similar to the those already presented in Sect. 3:

1. **Encryption** of one bit $b \in \{-1, 1\}$ is done generating a random value y relative prime with N; the ciphertext is $c = E(b) = y^2 x^{(1-b)/2} \pmod N$.
2. **Decryption** of a ciphertext c is done by computing the Legendre symbol (x/p) using the prime factorization (p, q): $m = D(c) = (\frac{x}{p})$.

This GM cryptosystem inherits homomorphic properties, in the sense that if c_0, c_1 are the encryptions of bits $m_0, m_1 \in \{-1, 1\}$, then $c_0 c_1 \bmod N$ will be an encryption of $m_0 m_1$. We also need a homomorphic mapping $\chi : H \rightarrow (R, \cdot)$, which in our case is the identity application: $\chi(h) = h$. From Sect. 5, it is easy to observe that the set S, the image of χ in R, is $\{-1, 1\}$.

6.1 Plaintext Decomposition

The next step from the general framework described in the previous section, is to find a fixed tuple $(r_1, \ldots, r_k) \in \mathbb{Z}_m^k$, where $k \geq 2$, such that the set containing elements of the form $\sum_{i=1}^{k} \pm r_i$ covers the whole ring R. One can observe that we have at most 2^k elements generated by the previous sums, therefore $2^k \geq N$.

Suppose that the binary representation of m requires B bits, i.e. $B = \lceil log_2(m) \rceil$. We propose the choose of the following parameters: $k = B$ and the tuple $(r_1, \ldots, r_k)$ as the set $\{2^0, 2^1 \ldots, 2^{B-1}\} \subset \mathbb{Z}_m$. First, let's consider an odd residue modulus m written in its binary representation: $r = \overline{b_{B-1} \ldots b_1 b_0}, b_i \in \{0, 1\}$. Now consider the subset I of indexes from $B - 1$ to 0 corresponding to non null bits: $I = \{i | B - 1 \geq i \geq 0, b_i = 1\} = \{i_1 > i_2 > \ldots > i_f\}$ with $f = |I|$. Clearly, since r is odd, $i_f = 0$. Then, we have:

$$r = \sum_{0 \leq i < B} b_i 2^i = \sum_{i \in I} b_i 2^i = 2^{i_1} + \ldots + 2^{i_f} \tag{22}$$

Considering each term 2^{i_k} from the previous sum, separately, then for $k > 1$, it can be written in the following way:

$$2^{i_k} = 2^l - \left(\sum_{l > j \geq i_k} 2^j \right), l = i_{k-1} - 1 \tag{23}$$

One can observe that if the indexes i_k and i_{k-1} are consecutive numbers, then the Eq. 23 still holds, the sum in the paranthesis disappearing completely. Moreover, for the most significative bit, non null, from r binary representation (i.e. $k = 1$), we have a similar formula:

$$2^{i_1} = 2^{B-1} - \left(\sum_{B-1>j\geq i_1} 2^j \right) \tag{24}$$

Unifying the formulas 22, 23 and 24, one can observe that any odd residue $r \in \mathbb{Z}_m$ can be written in terms of $\sum \pm 2^i \bmod m, 0 \leq i \leq B-1$, as follows:

$$r = \sum_{1\leq k\leq f} \left(2^l - \left(\sum_{l>j\geq i_k} 2^j \right) \right), l = \begin{cases} i_{k-1} - 1 & \text{if } k > 1 \\ B-1 & \text{if } k = 1 \end{cases}$$

To better understand this decomposition in terms of $\sum \pm 2^i$, let's consider an example: $m = 61$ and $r = 23 = \overline{010111}_2$. The set of r_i is $\{2^5, 2^4, 2^3, 2^2, 2^1, 2^0\}$. Then, we have: $23 = 2^4 + 2^2 + 2^1 + 2^0 = 2^5 - 2^4 + 2^3 - 2^2 + 2^1 + 2^0$. To conclude, any odd residue from $\mathbb{Z}_m$ can be written in terms of the chosen set.

Let's consider now the case of an even residue r from $\mathbb{Z}_m{}^*$. Then, $m - r$ is an odd residue and therefore can be written as: $m - r = \sum s_i 2^i$, with $s_i \in \{-1, 1\}$. That means that: $r - m = \sum(-s_i)2^i$. Reducing the two expressions modulus m, it gives us the following writing: $r = \sum(-s_i)2^i \bmod m$, which corresponds to the decomposition of r in terms of the chosen set. The decomposition of the residue 0 from $\mathbb{Z}_m$, can be obtained from the writting of m (an odd number) just like in the same manner as all other odd residues from $\mathbb{Z}_m$.

Therefore, we know at this moment, a precise method to write any residue from $\mathbb{Z}_m$ as a sum of the form $\sum s_i r_i$, for the chosen set of $r_i = 2^i$. We'll explain further the key generation, the encryption and the decryption processes which derives naturally from the general framework described in Sect. 5.

6.2 Key Generation, Encryption and Decryption

Key Generation consists in the generation of two distinct large prime numbers p and q, randomly and independently of each other. One computes then $N = pq$. Further, one finds some non-residue x such that the Legendre symbols satisfy $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$ and hence the Jacobi symbol $\left(\frac{x}{N}\right)$ is $+1$. The public key consists of (x, N), while the secret key is the factorization $N = pq$.

Encryption process is described by the following steps:

1. For a plaintext $r \in \mathbb{Z}_m$, one computes the tuple $(s_1, \ldots, s_B) \in \{-1, 1\}^k$ as described in Subsect. 6.1, such that: $r = \sum r_i s_i, 1 \leq i \leq B$.
2. For each $i \in \{1 \ldots B\}$, one generates a random value y_i relative prime to N and then encrypts s_i as $c_i = E(s_i) = {y_i}^2 x^{(1-s_i)/2} \pmod N$.

3. The encryption of the plaintext $r \in \mathbb{Z}_m$ is the following expression from the group algebra ring $\mathbb{Z}_m[\mathbb{Z}_N]$:

$$Enc(r) := \sum_{i=1}^{B} 2^{i-1}[c_i] \tag{25}$$

Decryption of an element from $\mathbb{Z}_m[\mathbb{Z}_N]$ is computed using the secret key (p, q) and is defined by the formula:

$$Dec\left(\sum_{c\in\mathbb{Z}_N} r_c[c]\right) := \sum_{c\in\mathbb{Z}_N} r_c D(c) = \left(\sum_{c\in\mathbb{Z}_N} r_c \left(\frac{c}{p}\right)\right) mod\ N \tag{26}$$

It is important to note that our scheme does NOT make a *bitwise encryption.* We can see that the plaintext space is Z_m and the encryption becomes homomorphic over both multiplicative and additive operations using the GM's multiplicative homomorphic properties. The GM scheme can be replaced within the above construction by any other encryption schemes, which have homomorphic properties with respect to the multiplication operation (e.g. Paillier encryption).

6.3 A Toy Example

To better understand the homomorphic encryption system based on GM scheme, let's consider a small example with the following parameters: $p = 7, q = 11$ and $N = pq = 77$. Therefore, N beeing a Blum number, i.e. $p \equiv q \equiv 3 \bmod 4$, we can choose $x = N - 1 = 76$; indeed, $(\frac{76}{7}) = (\frac{76}{11}) = -1$. The public key is the pair $(x = 76, N = 77)$ and the secret key is the factorization $(p = 7, q = 11)$.

Let's choose now $m = 7$, so the plaintext space is the ring $\mathbb{Z}_7$ and the B parameter from our scheme is $B = 3$. Suppose we want to encrypt two residues from $\mathbb{Z}_7$, namely 5 and 4. First, the decomposition of 5 is $5 = -1 + 2 + 4 \bmod 7$, so the set of coefficients s_i to be encrypted using GM is $\{-1, 1, 1\}$. Using the encryption algorithm described in Subsect. 6.2, we generates the 3 corresponding encryptions for $\{-1, 1, 1\}$ using the set of y_i as $\{2^2, 3^2, 5^2\}$; the encrypted values are $\{73, 9, 25\}$. Therefore, the encryption of 5 is as follows: $c_5 = Enc(5) = 1[73] + 2[9] + 4[25]$.

Second, the decomposition of 4 is the following: $4 = -3 = -1 + 2 - 4 \bmod 7$, so the set of coefficients s_i to be encrypted using GM is $\{-1, 1, -1\}$. Using the encryption algorithm described in Subsect. 6.2, we generates the 3 corresponding encryptions for $\{-1, 1, -1\}$ using the set of y_i as $\{4^2, 5^2, 1^2\}$; the encrypted values are $\{61, 25, 76\}$. Therefore, the encryption of 4 is as follows: $c_4 = Enc(4) = 1[61] + 2[25] + 4[76]$.

Now let's compute $c_4 + c_5$ and c_4c_5 within the ciphertext space. In the next formulas we used the equations describing the group algebra operations from Sect. 4 together with the online tool [13] for computing Legendre symbols.

$$\begin{aligned} c_4 + c_5 &= (1[73] + 2[9] + 4[25]) + (1[61] + 2[25] + 4[76]) \\ &= 1[73] + 2[9] + (4 + 2 \ mod\ 7)[25] + 1[61] + 4[76] \\ \boldsymbol{Dec(c_4 + c_5)} &= 1\left(\frac{73}{7}\right) + 2\left(\frac{9}{7}\right) + 6\left(\frac{25}{7}\right) + 1\left(\frac{61}{7}\right) + 4\left(\frac{76}{7}\right) \ mod\ 7 \\ &= (-1 + 2 + 6 - 1 - 4) \ mod\ 7 = 2 = \boldsymbol{5 + 4 \ mod\ 7} \end{aligned}$$

$$\begin{aligned} c_4c_5 &= (1[73] + 2[9] + 4[25])\,(1[61] + 2[25] + 4[76]) \\ &= [73 \cdot 61] + 2[73 \cdot 25] + 4[73 \cdot 76] + 2[9 \cdot 61] + 4[9 \cdot 25] + [9 \cdot 76] \\ &\quad +4[25 \cdot 61] + [25 \cdot 25] + 2[25 \cdot 76] \\ &= [64] + 2[54] + 4[4] + 2[10] + 4[71] + [68] + 4[62] + [9] + 2[52] \\ \boldsymbol{Dec(c_4c_5)} &= \left(\frac{64}{7}\right) + 2\left(\frac{54}{7}\right) + 4\left(\frac{4}{7}\right) + 2\left(\frac{10}{7}\right) + 4\left(\frac{71}{7}\right) \\ &\quad + \left(\frac{68}{7}\right) + 4\left(\frac{62}{7}\right) + \left(\frac{9}{7}\right) + 2\left(\frac{52}{7}\right) \ mod\ 7 \\ &= (1 - 2 + 4 - 2 + 4 - 1 - 4 + 1 - 2) \ mod\ 7 = 6 = \boldsymbol{5 \cdot 4 \ mod\ 7} \end{aligned}$$

7 Implementation and Experimental Results

The HE-GM is our implementation of the homomorphic encryption system presented in Sect. 6 of the paper. It has been written in C++ and is based on the NTL mathematical library [14]. The code includes the routines for GM scheme (GM-KeyGen, GM-Enc, GM-Dec) and the implementation of the homomorphic encryption system over group algebras (as described in Sect. 6.2).

The HE-GM can encrypt integer values of any B-bits lengths and get a fresh ciphertext with B terms each of them containing a GM encryption of one bit. The two basic homomorphic operations (addition and multiplication) have been implemented in the HE-GM at the ciphertext level. Using the HE-GM implementation we validated the correctness of the homomorphic encryption system. We made also various benchmarks that aim for time consumption necessary to achieve fresh data encryption/decryption, evaluation of add and multiply operations and the ciphertext sizes. The benchmarks have been carried out using different security levels for GM scheme (various sized key-parameters p, q).

Our experiments were conducted on a normal laptop having an Intel CPU (I7-4710HQ, 4 cores, 2.5 GHz, 3 GB RAM). The implementation is not multithreaded and it uses only one CPU core. The Table 1 presents the costs in terms of time and ciphertext size needed by a fresh encryption and decryption of an integer value with a binary representation length of 8 bits.

The Table 2 contains computation time measured during the evaluation of basis operations (adding and multiplying). The most time consuming operation is the multiplication, because in that case the number of terms from resulting ciphertext is the sum of terms contained by evaluated ciphertexts. We note that the growth factor for time spent for each additional multiplication with a fresh encrypted value is kept approximately constant.

Table 1. Fresh encryption and decryption of an integer value using HE-GM system

GM key-params $p,\ q$	Enc. time	Dec. time	Ciphertext size
$p, q = 1024$ bits	3.23 ms	0.8 ms	2072 bytes
$p, q = 2048$ bits	10 ms	2.3 ms	4120 bytes
$p, q = 4096$ bits	40 ms	6.5 ms	8216 bytes

Table 2. Time costs for HE-GM homomorphic operations

GM key-params p, q	$a + b$	$a * b$	$a * b * c$	$a * b * c * d$	$a * b * c * d * e$
$p, q = 1024$ bits	0.07 ms	0.8 ms	7.85 ms	64 ms	770 ms
$p, q = 2048$ bits	0.11 ms	2.205 ms	21 ms	163 ms	1637 ms
$p, q = 4096$ bits	0.15 ms	6.7 ms	67 ms	500 ms	4.5 s

Table 3 presents a comparison between our HE scheme implementation over GM (HE-GM) and the leveled implementation of HElib [10]. We used a 2048 bit length for the GM key. The values are calculated as an average execution time consumed by the implementation for multiplying integers of various length. The results show that for the case of small integers, our HE-GM system is considerable faster than HElib. Using the leveled variant of HElib, the time consumption in its case is relative constant. In the case of HE-GM, the number of multiplication operations has a polynomial growth for each additional multiplication.

Table 3. Timing costs for HE-GM and HElib in case of multiply operations

Number of bits	$a * b$		$a * b * c$		$a * b * c * d$		$a * b * c * d * e$	
	HE-GM	HElib	HE-GM	HElib	HE-GM	HElib	HE-GM	HElib
8 bits	0.8 ms	347 ms	7.85 ms	870 ms	64 ms	1 542 ms	770 ms	2 269 ms
16 bits	3.4 ms	336 ms	60 ms	851 ms	2193 ms	1 503 ms	510 s	2 374 ms
24 bits	7.8 ms	334 ms	241 ms	846 ms	44 060 ms	1 451 ms	107 min	2 205 ms

8 Conclusion

This paper builds on a general framework able to extend a group homomorphic encryption scheme with respect to one operation, towards a ring homomorphic cryptosystem. This new cryptosystem has homomorphic properties on two operations: addition and multiplication. We choose to apply the general framework to a well known homomorphic encryption scheme, Goldwasser-Micali, and analyze the resulted cryptosystem from the security and the efficiency point of view.

The security of the proposed scheme is the same as the security of the initial group encryption scheme (i.e. Goldwasser-Micali) since no information and no additional security was revealed or added through the steps describing the encryption process as described previously in Sect. 5.2. The GM cryptosystem is semantically secure based on the assumed intractability of the quadratic residuosity problem corresponding to a modulus product of two large large primes.

From the efficiency point of view, as illustrated by the experimental results, our scheme works well for the case of small integers (byte values) but shows its weakness for large integers, especially when the number of multiplications grows up. This is basically due first to the expansion introduced by Goldwasser-Micali on a bit level and second (more important) by the expansion given by operations on ciphertexts. As shown previously, the parameter k (i.e. the number of bits) has a direct (linear) impact over the length of fresh ciphertexts and the addition operation, while in the multiplication process the length of ciphertext will grow up to the product of the ciphertexts' lengths.

Therefore, one important perspective of our work regards the application of the general framework on schemes having smaller groups (i.e. smaller k) that contains the result of the encryption process. Another perspective concerns the application of the general framework to other encryption schemes known as group homomorphic schemes like RSA, ElGamal, Paillier, Diffie-Hellman, etc.

The blueprint of the above described encryption scheme opens the path of constructing new families of secure ring/fully-homomorphic encryption schemes which are NOT error-based. The efficiency issues are of different nature than those of error-based encryption schemes, and further improvements might bring better understanding of how far one can go in the attempt of realizing practical fully homomorphic encryption schemes.

Acknowledgments. This research was partially supported by the Romanian National Authority for Scientific Research (CNCS-UEFISCDI) under the project PN-II-PT-PCCA-2011-3 (ctr. 19/2012).

References

1. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–179. Springer, Academia Press (1978)
2. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009). http://crypto.stanford.edu/craig
3. Barcău, M., Paşol, V.: Fully Homomorphic Encryption from Monoid Algebras (2016)
4. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984). Massachusetts Institute of Technology, Cambridge
5. Fellows, M., Koblitz, N.: Combinatorial cryptosystems galore! In: Finite Fields: Theory, Applications, and Algorithms. Contemporary Mathematics, vol. 168, pp. 51–61. AMS (1994)
6. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: Innovations in Theoretical Computer Science Conference, pp. 309–325 (2012)

8. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Canetti, R., Safavi-Naini, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
9. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Crypt. **71**, 57–81 (2012)
10. Halevi, S., Shoup, V.: The HElib library (2015). https://github.com/shaih/HElib
11. Grigoriev, D., Ponomarenko, I.: Homomorphic public-key cryptosystems over groups and rings. Quad. di Math. **13**, 305–325 (2004)
12. Ireland, K., Rosen, M.: A Classical Introduction to Modern Number Theory, 2nd edn. Springer, New York (2000)
13. Richman, F.: http://math.fau.edu/richman/jacobi.htm
14. Shoup, V.: NTL: A library for doing number theory (2001)