Primeless Factoring-Based Cryptography

–Solving the complexity bottlenecks of public-key encryption with ephemeral keys–

No Author Given

No Institute Given

Abstract. Factoring-based public-key cryptosystems have an overall complexity which is dominated by the key-production algorithm, which requires the generation of prime numbers. This is most inconvenient in settings where the key-generation is not an one-off process, e.g., secure delegation of computation or EKE password-based key exchange protocols. To this end, we extend the Goldwasser-Micali (GM) cryptosystem to a provably secure system, denoted *SIS*, where the generation of primes is bypassed. By developing on the correct choice of the parameters of SIS, we align SIS's security guarantees (i.e., resistance to factoring of moduli, etc.) to those of other well-known factoring-based cryptosystems. Taking into consideration different possibilities to implement the fundamental operations, we explicitly compare and contrast the asymptotic complexity of well-known public-key cryptosystems (e.g., GM and/or RSA) with that of SIS's. The latter shows that once we are ready to accept an increase in the size of the moduli, SIS offers a generally lower asymptotic complexity than, e.g., GM or even RSA (when scaling correctly the number of encrypted bits). This would yield most significant speed-ups to applications like the aforementioned secure delegation of computation or protocols where a fresh key needs to be generated with every new session, e.g., EKE password-based key exchange protocols.

1 Introduction

Setting. Several, widely used public-key cryptosystems have a setup phase where prime numbers are generated and/or primality tests are run. The computational complexity yielded by the generation of a prime number of length L is generally in $O(L^4)$ and –if optimised– $O^{\sim}(L^3)$, as we will detail next in this section. Such generations occur, for instance, in the case of RSA [22] and/or in the Goldwasser-Micali (GM) probabilistic cryptosystem [8], as each of them defines its operation over $\mathbf{Z}_n^*$, for n being a product of two, distinct large prime numbers generated therein.

Moreover, there exist settings in which the key-generation in asymmetric cryptosystems is not an one-off process. Such a case is that of the increasingly popular of delegation of computation, where some client outsources a task, e.g, the solving a linear system, to a remote worker. *Secure* delegation protocols [19] are based on homomorphic public-key encryption schemes and in each of the runs of such a protocol, the keys need to be re-issued freshly. Hence, the asymptotic complexity of prime-generation for the homomorphic encryptions used therein [19] (e.g., GM, RSA, Paillier's encryption, encryption based on bilinear maps, etc.) becomes an alarming bottleneck of the delegated computation. The scheme that we propose in this paper is homomorphic and it is aimed at overcoming precisely the shortcoming of such bottlenecks. In this context, in our comparisons, we focus mostly on (homomorphic) schemes that are commonly used in these settings i.e., factoring-based ones, and do not compare with public-key cryptosystems different in nature. I.e., we do not refer to the McEliece cryptosystem [18] based on algebraic codes, which may indeed have faster key-generation procedures; neither do we related to Diffie-Hellman cryptosystems (e.g., EC-based) for which primes are not key-specific. Such systems are briefly discussed in Appendix C. Again, we focus on cryptosystems using some primes which are key-specific. We therefore focus on such public-key cryptosystems in which the key-generations require primality testing and where the security gravitates around problems related to primality, e.g., factoring of moduli. More precisely, we extend and compare with the Goldwasser-Micali (GM) probabilistic cryptosystem [8].

Along with secure delegation of computation, there are other cases in public-key cryptography where the key-generation are not an one-off process is in the settings of 1. EKE password-based key exchange [4]; 2. zero-knowledge proofs of knowledge without any setups, where a new commitment key is used at each session; 3. key agreement with forward secrecy [1]. Thus, we believe that our study is founded on valid concerns.

Comparisons at a glance. The security of the RSA cryptosystem is based on is the integer prime-factorisation problem (i.e., the RSA modulus n should be hard-to-factor). Thus, a fair security guarantee is to take the length L of the modulus n large enough to be considered practically hard-to-factor using, e.g., the general number field sieve (GNFS) factorisation algorithm [15]. Looking at the complexity of the latter factorisation for a number of the order 2^L and measuring its hardness in the order of 2^s (where s is a security parameter), it is to conclude that a secure length L for the RSA-generated modulus is of the order of $L = O^{\sim}(s^3)$ due to the complexity of the GNFS algorithm.

The commonplace implementation of the RSA cryptosystem has a complexity of $O(L^4)$ for the setup phase, due to prime-generation numbers. And, in general, this is the final complexity of the scheme, since the encryption and/or decryption processes run in only $O(L^3)$. The "schoolbook" multiplication method of $O(L^2)$ can be replaced by "fast multiplication" techniques in the key-generation process, i.e., by the Karatsuba algorithm [12] in $O(L^{\log_2 3})$ or by methods [23] based on the Fast Fourier Transform (FFT) in $O(L \times \log L)$. In the FFT-based optimisation cases, the complexity of RSA is lowered[1] to $O^{\sim}(L^3)$. This cannot be further improved as primality generating and testing itself does not come cheap. The prime number theorem states that the probability for an integer randomly selected in the vicinity of some large integer n to be prime is about $\frac{1}{L}$, where L is the bitlength of n. In the FFT-optimised case, the Miller-Rabin primality-testing drops from its standard $O(L^3)$ as far as to an overall complexity of $O(L^2 \log L \log\log L) = O^{\sim}(L^2)$. For the state-of-the-art deterministic primality-testing method (AKS [3]), the complexity has been shown [17] to be of the order of $O(L^6)$. State-of-the-art, reliable probabilistic tests (elliptic curve primality proving) [14] are computational expensive as well, i.e., $O(L^{5+\varepsilon})$. Thus, in the best case of FFT-optimisation, primality-testing based cryptosystems would run in $O^{\sim}(s^9)$, whilst commonly they would run in $O(s^{12})$.

The GM cryptosystem is semantically secure under the assumption that the quadratic residuosity (QR) problem modulo a composite integer n is hard. As in the case of RSA, this modulus n is obtained as the product of two distinct, freshly generated odd prime numbers. The QR problem stipulates that, given this modulus n and a number $x \in \mathbf{Z}_n^*$, when the Jacobi symbol [11] for $x \in \mathbf{Z}_n^*$ with respect to n is 1, it is difficult to determine whether x is a quadratic residue modulo n (i.e., whether x equals $y^2 \bmod n$, for some $y \in \mathbf{Z}_n^*$). If the prime-factorisation of n is known, then the QR problem is easy. In this context of complexity-analysis, it is to be mentioned that the Jacobi symbol [11] itself generally has quadratic complexity, as schoolbook multiplication is most often used within. We briefly recall the GM scheme somewhat more detailedly. In the key-generation algorithm, firstly two different (large) prime numbers p and q are independently generated. Then, the modulus n is computed as pq. Then, a non-residue x is found such that its Legendre symbol [11] with respect to p and q are equal to -1, i.e., $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$, whereas the Jacobi symbol with respect to n is 1, i.e., $\left(\frac{x}{n}\right) = 1$. The public key is defined by the pair (x,n), whereas the prime factors p and q are kept secret. To encrypt a bit b, an integer y is randomly picked from $\mathbf{Z}_n^*$, i.e., $y \leftarrow_U \mathbf{Z}_n^*$, and its encryption is calculated as $c = y^2 x^b \pmod N$. To decrypt, the secret key (p,q) is used and it is to check whether the encrypted value c is a quadratic residue, i.e., to solve $\left(\frac{c}{p}\right) = (-1)^b$, in the unknown b. Note that if a polynomial adversary tries to break the cryptosystem on an input x that is a quadratic residue modulo n, then this adversary

[1] $O^{\sim}(t(n))$ is equal to $O(t(n) \times (\log t(n))^c)$, for some constant $c \geq 0$.

cannot output correctly in more than half of its trials (i.e., in answering whether x is indeed a quadratic residue). In other words, such an adversary must solve the QR problem to be successful in his attack.

Contribution. In this paper, we endeavour in extending the GM scheme into a public-key scheme that bypasses prime-generation procedures. The GM scheme has the same aforementioned complexity bottlenecks. We show reduction in complexity, from the usual $O^{\sim}(s^{12})$ to $O^{\sim}(s^{7.5})$, at the cost of generating larger, composite numbers (where s is the security parameter). This comes to the special benefit of applications like the aforementioned (e.g., secure delegation of computation, EKE password-based key exchange protocols, etc.), where the overwhelming key-generations repeat at each run.

Structure of the paper. In Section 2, we present the computational problems relevant to our cryptosystem and discuss their hardnesses. In Section 3, we describe our cryptosystem denoted *SIS*. In Section 4, we discuss the necessary conditions for the selection of parameters and their asymptotic behaviours. In Section 5, we discuss the complexity of our scheme and compare it to RSA and GM. We provide experimental results that show that our asymptotic analysis is valid.

2 Preliminaries

In this section, we present some essential background to the work herein, the computational problems that found the basis of our security analyses, and results with respect to their hardness.

2.1 Foundations

Let G be an Abelian group. A *character* χ is a group homomorphism from $(G,+)$ to $\mathbb{C}^*$. The set of characters over G has a group structure with component-wise multiplication over $\mathbb{C}^*$. This group is called the *dual group* $\widehat{G}$ of G and it is isomorphic with G [11]. Each character will have an order in this group. For all $a \in G$, $\chi(a)$ is a $\lambda(G)$-th root of the unity, where $\lambda(G)$ is the exponent of the group G. A character χ of order 2 is such that $\chi(a) \in \{-1,1\}$, for all $a \in G$. Let ε be the trivial character, i.e., $\varepsilon(a) = 1$. The set of characters χ for which $\chi^2 = \varepsilon$ consists of ε and characters of order 2.

Let $p \in \mathbf{Z}$ be an odd prime. The only character in $\mathbf{Z}_p^*$ of order 2 is the Legendre symbol $\chi(a) = (\frac{a}{p})$, for any $a \in \mathbf{Z}_p^*$. For the standard properties of the Legendre symbol, as well as its generalisation to the Jacobi symbol w.r.t. composed numbers, see [11]. For $n = pq$ with p and q being two different odd primes, there are 3 characters of order 2: the Legendre symbol $(\frac{\cdot}{p})$, the Legendre symbol $(\frac{\cdot}{q})$, and the Jacobi symbol $(\frac{\cdot}{n})$. The latter is easy to compute, but the former are allegedly hard to compute when the primes p and q are unknown. We call these former characters *hard characters* of order 2.

We recall that QR_n is a usual notation for the subgroup of Z_n^* of all quadratic residues. We refer to the problem of deciding whether an element of Z_n^* is quadratic residue or not as the QR problem.

The main scope of this paper is to use characters of order 2, in order to design public-key encryption schemes that elude the generation of prime numbers, thus reducing the general asymptotic complexity of the usual schemes of the kind.

2.2 Computational Problems

In this paper, we first consider the following combinatorial problem:

CHI Problem (Character Interpolation Problem):
Parameters: a modulus n, $x_1,\ldots,x_t$ in $\mathbf{Z}_n^*$, t elements $y_1,\ldots,y_t \in \{-1,+1\}$, all defining a unique character χ on $\mathbf{Z}_n^*$ such that $\chi(x_i) = y_i$ for $i = 1,\ldots,t$ and $t \geq 1$.

Input: $x \in \mathbf{Z}_n^*$.
Problem: Find $y = \chi(x)$.

An instance of this problem is defined by fixing some parameters and providing a corresponding input x. Then, it requires the computation of a character of order 2 for a number $x \in \mathbf{Z}_n^*$, given t elements in $\mathbf{Z}_n^*$ and their respective characters. Note that the above problem can be generalised to the case of characters of order d, i.e., by replacing the set $\{-1,+1\}$ with a group of order d. Also, observe the **CHI** problem can be immediately rewritten as the **MOVA^2** problem presented in [20].

We now give a combinatorial problem presented in [21]:

GHI Problem (Group Homomorphism Interpolation Problem):
Parameters: G and H two Abelian groups, S be a subset of $G \times H$, $r \geq 1$ such that S-interpolates in a homomorphism between G and H.
Input: r elements $x_1, \ldots, x_r$ in G.
Problem: Find $y_1, \ldots, y_r \in H$ such that there exists a group homomorphism φ such that $\varphi(x_i) = y_i$ for $i = 1, \ldots, r$ and $\varphi(x) = y$ for all $(x,y) \in S$.

An instance of the above problem demands that once given r numbers lying in G, one provides r points in H that together with S interpolate in the group homomorphism φ.

Obviously, the **CHI** problem is a specialisation of the **GHI** problem in which $G = \mathbf{Z}_n^*$, $H = \{-1,+1\}$, $r = 1$, and the homomorphism is unique.

We recall the following theorem:

Lemma 1. *(Lemma 4.3 in [21]) Let G and H be two finite Abelian groups, where the group operation is denoted additively. We denote by d the order of H. Let $x_1, x_2, \ldots, x_r \in G$ which span G'. The following properties are equivalent. In this case, we say that $x_1, x_2, \ldots, x_r$ H-generate G.*

- *For all $y_1, y_2, \ldots, y_r \in H$, there exists at most one group homomorphism $Hom : G \to H$ such that $Hom(x_i) = y_i$ for all $1 \leq i \leq r$.*
- *$G' + dG = G$.*
- *$x_1 + dG, \ldots, x_r + dG$ span G/dG.*

We denote $span_d(x_1, \ldots, x_t) = \langle x_1, \ldots, x_t \rangle + dG$. Then, saying that $\{x_1, \ldots, x_t\}$ H-generates for $H = \{-1,+1\}$ is equivalent to $\mathsf{span}_2(x_1, \ldots, x_t) = \mathbf{Z}_n^*$.

When one can compute discrete logarithms in $\mathbf{Z}_n^*$, one can easily solve the **CHI** problem by solving a linear system. The discrete logarithm is easy when n has only small prime factors. Therefore, for the **CHI** problem to be hard, we need that n has large prime factors. This is the case if n is hard-to-factor. Similarly, when n is easy to factor, we can easily evaluate the characters in certain subgroups of $\mathbf{Z}_n^*$, and therefore solve the **CHI** problem. For details, an efficient Karp reduction of the **CHI** problem to the factorisation problem is present in [20].

Definition 2 (CHI and QR Hardness Assumptions). *Given a probabilistic algorithm $\mathsf{Gen}(1^s) \to (n, \chi, t)$ such that χ is a character of order 2, say that the **CHI** problem is* hard *relative to* Gen *if for every probabilistic algorithm $\mathcal{A}$ which is polynomial in s, we have*

$$\left| \Pr \left[\mathcal{A}(x, n, x_1, \ldots, x_t, \chi(x_1), \ldots, \chi(x_t)) = \chi(x) \;\middle|\; \begin{array}{l} \mathsf{Gen}(1^s) \to (n, \chi, t), \\ x, x_1, \ldots, x_t \in_U \mathbf{Z}_n^*, \\ \mathsf{span}_2(x_1, \ldots, x_t) = \mathbf{Z}_n^* \end{array} \right] - \frac{1}{2} \right| < \mathsf{negl}(s).$$

We say that the QR problem is hard *relative to* Gen *if for every probabilistic algorithm $\mathcal{A}$ which is polynomial in s, we have*

$$\left| \Pr \left[\mathcal{A}(x,n) = 1_{x \in QR_n} \middle| \begin{array}{l} \mathsf{Gen}(1^s) \to (n), \\ x \in_U \mathbf{Z}_n^*, \\ \left(\frac{x}{n}\right) = 1 \end{array} \right] - \frac{1}{2} \right| < \mathsf{negl}(s).$$

Then, we have the following amplification result.

Lemma 3. *Given the parameters n, $x_1, \ldots, x_t, y_1, \ldots, y_t$ for an instance of the **CHI** problem, i.e., defining a unique χ on $\mathbf{Z}_n^*$, if $\mathcal{A}$ is a probabilistic algorithm which is polynomial in s such that*

$$\left| \Pr \left[\mathcal{A}(x) = \chi(x) \,|x \in_U \mathbf{Z}_n^* \right] - \frac{1}{2} \right| > \theta,$$

then one can define an algorithm $\mathcal{A}'$ calling $\mathcal{A}$ a number of $\frac{1}{2}\theta^{-2} \ln \frac{2}{\varepsilon}$ times such that

$$\Pr \left[\mathcal{A}'(x) = \chi(x) \,|x \in_U \mathbf{Z}_n^* \right] \geq 1 - \varepsilon,$$

with $\theta > 0$ and $\varepsilon > 0$.

Due to space constraints, we present the proof of the above in the appendix.

Now assume a ppt. algorithm Gen_{GM} which generates a modulus $n = pq$ as in the Goldwasser-Micali cryptosystem [8] and $t > 1$, i.e., $\mathsf{Gen}_{GM}(1^s) \to (n)$. We define $Gen_{CHI}(1^s) \to (n, (\frac{\cdot}{p}), t)$, given that we have $\mathsf{Gen}_{GM}(1^s) \to (n)$, p being one of the two prime factors of n selected at random, and $t = 2$. We can then see that the hardness of quadratic residuosity implies the hardness of the **CHI** problem relative to Gen_{GM}. Formally, this is proven below.

Theorem 4. *If the QR problem is hard relative to Gen_{GM}, then **CHI** problem is hard relative to Gen_{CHI}.*

Due to space constraints, we present the proof of the above in Appendix A, page 15.

In this paper, we use the CHI problem with $\chi(\cdot) = \left(\frac{\cdot}{\alpha}\right)$ over $\mathbf{Z}_n^*$, for a factor α of n. For the CHI problem to be hard, α must be a hard factor of n. So, we tune our parameters to ensure this. So far, we know no algorithm better than finding α to solve the CHI problem. So, we believe that our selection method is enough to guaranty security.

3 `SIS`: A Primeless Public-Key Cryptosystem

Our proposed scheme, denoted `SIS`, is described below. We assume a security parameter s. Based on s, other parameters of `SIS` will be defined: t, k, and ℓ. The exact way to choose these parameters (in order to ensure the security of `SIS`) is discussed in Section 4.

3.1 The Core of the Cryptosystem

In Algorithm 1, we describe the *key generation procedure* of our `SIS` cryptosystem. As usual, the procedure runs in the security parameter s. Algorithm 1 generates and uses within a parameter denoted t, which will have its expression in this security parameter s made clear in the next section.

This key generation procedure produces a pair (α, n) of integers such that α is an odd factor of n. We note that the value n is part of the public key, whereas the integer α is kept secret. Therefore,

the Jacobi symbol $(\frac{\cdot}{\alpha})$ is a character of order 2 in $\mathbf{Z}_n^*$. Then, in the generation procedure, t values, $x_1, x_2, \ldots, x_t$, are randomly picked from $\mathbf{Z}_n^*$. Using the Jacobi symbol $(\frac{\cdot}{\alpha})$, the values y_i are computed as $(\frac{x_i}{\alpha})$, for all $1 \leq i \leq t$. If all y_i's are equal to 1, the all the x_i values are dropped and the procedure restarts by choosing again all these values, in the same fashion. (In any case, this occurrence of re-starting is rare: it happens with a probability close to 2^{-t}; there are rare cases where all y_i's are always 1, i.e., when α is a square.)

1: **Input:** Security parameter s.
Output: Public key: $(n, x_1, x_2, \ldots x_t, y_1, y_2, \ldots y_t)$; Private key: α.

2: compute t, k, and ℓ depending on s, as per (1) in p. 9 and (2)-(4) in p. 10
3: pick random odd integers α_i and β_i of size ℓ, $i = 1, \ldots, k$;
4: compute $\alpha = \alpha_1 \times \cdots \times \alpha_k$
5: compute $\beta = \beta_1 \times \cdots \times \beta_k$
6: compute $n = \alpha \cdot \beta$
7: pick $x_1, x_2, \ldots, x_t \in_U \mathbf{Z}_n^*$
8: compute $y_i = (\frac{x_i}{\alpha})$ for all $1 \leq i \leq t$
9: **if** $y_i = 1$ for all $1 \leq i \leq t$, **then** go-to step 3

Algorithm 1: `SIS`: `Key generation`

Intuitively, we could expect that taking $k = 1$ would be the optimal option. We will see in the analysis in the next section that there is an advantage in taking k larger.

In Algorithm 2 below, we show how to encrypt a bit b using our `SIS` cryptosystem. From the

1: **Input:** a bit b.
Public key: $(n, x_1, x_2, \ldots x_t, y_1, y_2, \ldots y_t)$.
Output: the encryption z, $z \in \mathbf{Z}_n^*$.

2: find $y_i = -1$, $i \in \{1, \ldots, t\}$
3: pick $b_1, b_2, \ldots, b_{i-1}, b_{i+1}, \ldots, b_t \in_U \{0, 1\}$
4: compute $P = \prod_{j \neq i} y_j^{b_j}$
5: **if** $P = (-1)^b$ **then** $b_i \leftarrow 0$ **else** $b_i \leftarrow 1$.
6: compute $z' = x_1^{b_1} \cdots x_t^{b_t} \pmod{n}$
7: pick $r \in_U \mathbf{Z}_n^*$
8: compute $z = r^2 \cdot z' \pmod{n}$

Algorithm 2: `SIS`: `Encryption`

public values y_i, the encryption procedure firstly selects t bits such that $\prod_i y_i^{b_i} = (-1)^b$. The value $z = r^2 \times x_1^{b_1} \cdots x_t^{b_t} \pmod{n}$ is computed, where the number r is randomly picked. The result denotes the ciphertext of the bit b. Having got the ciphertext z and knowing value α, one decrypts z by solving $(-1)^b = (\frac{z}{\alpha})$, where $(\frac{z}{\alpha})$ is the Jacobi symbol of z with respect to α. This is presented in Algorithm 3.

Correctness. The encryption scheme the `SIS` cryptosystem is correct, i.e., if z is the encryption of a bit b as above, then the participant A decrypts z to b, provided that he knows the secret value α. To see this, we give the following lemma.

1: **Input:** the encryption z, $z \in \mathbf{Z}_n^*$.
Secret key: α.
Output: a bit b.

2: compute $(\frac{z}{\alpha})$.
3: **if** $(\frac{z}{\alpha}) = 1$ **then** $b = 0$ **else** $b = 1$.

Algorithm 3: SIS: Decryption

Lemma 5. *$(\frac{z}{\alpha}) = (-1)^b$, where the values z, α, the bit b are honestly computed/selected as in the SIS cryptosystem.*

Proof. From $z = r^2 x_1^{b_1} \cdots x_t^{b_t} \pmod{n}$ as in the scheme and α a divisor of n, it follows that $z = r^2 x_1^{b_1} \cdots x_t^{b_t} \pmod{\alpha}$. Using the standard properties for the Legendre symbol, we obtain that the value $(\frac{z}{\alpha})$ that A finally calculates is as follows:

$$(\tfrac{z}{\alpha}) = (\tfrac{r^2 x_1^{b_1} \cdots x_t^{b_t}}{\alpha}) = (\tfrac{r}{\alpha})^2 (\tfrac{x_1}{\alpha})^{b_1} \cdots (\tfrac{x_t}{\alpha})^{b_t} = 1 \cdot y_1^{b_1} y_2^{b_2} \cdots y_t^{b_t} = (-1)^b. \qquad \square$$

3.2 Security Analysis

It is clear that to perform a secret key recovery attack, an attacker needs to find the factor α of n. So, SIS strongly relies on the factoring problem.

Take now the goal of the adversary to be guessing whether b is 0 or 1. We follow the standard lines in saying that our cryptosystem is IND-CPA secure if for every polynomial adversary A outputting a bit b', its an advantage $Adv_{\mathcal{A}}(s) = \Pr_{\alpha,n,X,B}[b = b'] - \frac{1}{2}$ is negligible in terms of s, where n and α are the modulus used and its secret factor generated in the scheme, b is the encrypted bit, X and B respectively denote the values x_i and b_i that are picked during a run of the scheme.

We easily conclude that the scheme is semantically secure (i.e., IND-CPA secure).

Let a ppt. algorithm *Gen* such that $\mathsf{Gen}(1^s) \to (n, \chi, t)$ with $\chi(\cdot) = \left(\frac{\cdot}{\alpha}\right)$ as per our system.

Corollary 6. *Assuming that the **CHI** problem is hard relative to Gen, the SIS scheme is IND-CPA secure.*

Proof. It follows from the definition of hardness of the **CHI** problem in Section 2.2, Lemma 3 and the construction of the SIS scheme, i.e., the bit that an $\mathcal{A}$ is supposed to output correctly to break IND-CPA security is the character $(\frac{z}{\alpha})$, where z is generated in Algorithm 2 and α is the secret generated at the setup phase. $\square$

Thus, we reduced the IND-CPA security of the SIS scheme to the hardness of the **CHI** problem (which is assumed to be hard). As aforementioned, the next sections expand on the hardness of the factorisation problem for n, a modulus generated as in the SIS scheme.

Since the SIS-scheme is homomorphic, it is clearly not IND-CCA secure.

4 Selection of the Parameters

In the first part of this section (i.e., in Subsection 4.1), let us assume that the parameters k and ℓ are chosen and we attempt to gauge the right choice for t.

4.1 The Local Parameter t

Let $s \in \mathbf{Z}$ be the security parameter and $L = 2k\ell$ be the bitlength of n. We pick the value t such that we obtain the uniqueness of the homomorphism in the **GHI** corresponding problem. Namely, we pick t to be greater than the value r specified by Lemma 1, specialised here for $d = 2$.

We state the following result from [21].

Theorem 7. *(Theorem 4.29 in [21]) Let G, H be some Abelian groups, and d the order of H. The probability P_{gen} that some elements $g_1, \ldots, g_s \in_U G$ picked uniformly at random H-generate G satisfies*

$$P_{\mathsf{gen}} \geq \prod_{q \in \mathcal{P}_d} \left(1 - \frac{q^{k_q} - 1}{(q-1) \cdot q^s}\right),$$

where $\mathcal{P}_d$ is the set of all prime factors of $\gcd(\#G, d)$ *and k_q is the rank of the maximal q-subgroup of G. (Given a prime q, the q-subgroup of G is the subgroup A_q of elements whose order are powers of q. The rank k_q is the integer such that there exists a unique sequence of integers $a_{q,1} \leq \cdots \leq a_{q,k_q}$ such that A_q is isomorphic to $\mathbf{Z}_{q^{a_{q,1}}} \oplus \cdots \oplus \mathbf{Z}_{q^{a_{q,k_q}}}$).*

To apply the above theorem to our case, we give the following corollary.

Corollary 8. *The probability that $\{x_1, \ldots, x_t\}$ in the* `SIS` *scheme $\mathbf{Z}_2$-generates $\mathbf{Z}_n^*$ is*

$$P_{\mathsf{gen}} \geq 1 - 2^{k_2 - t},$$

where k_2 is the rank of the group A_2 and A_2 is the maximal 2-subgroup of $\mathbf{Z}_n^*$.

In order to enforce that $1 - P_{\mathsf{gen}}$ is smaller than 2^{-s}, we get a sufficient bound for t: i.e., $t \geq k_2 + s$.

Further, the rank k_2 of the 2-subgroup of $\mathbf{Z}_n^*$ is closely related to $\omega(n)$, i.e., the number of distinct prime factors of n [9]. More precisely the relation is as follows.

Lemma 9. *The rank k_2 of the 2-subgroup of $\mathbf{Z}_n^*$ is:*

$$\begin{cases} \omega(n), & \text{if } n \text{ is odd or } (4 \text{ divides } n \text{ and } 8 \text{ does not divide } n) \\ \omega(n) - 1, & \text{if } 2 \text{ divides } n \text{ and } 4 \text{ does not divide } n \\ \omega(n) + 1, & \text{if } 8 \text{ divides } n \end{cases}$$

Proof. We write n as $\prod_{i=1}^{r} p_i^{\alpha_i} \times 2^{\alpha_0}$, where p_i are different, odd primes. Then, by properties of Abelian groups, $\mathbf{Z}_n^*$ is isomorphic with the group $\prod_{i=1}^{r} \mathbf{Z}_{p_i^{\alpha_i}}^* \times \mathbf{Z}_{2^{\alpha_0}}^*$. The group $\mathbf{Z}_{p_i^{\alpha_i}}^*$ is cyclic of 2-rank equal to 1, for each p_i as above. The group $\mathbf{Z}_{2^{\alpha_0}}^*$ is: either the trivial group, hence of 2-rank equal to 0 (if $\alpha_0 = 0$ or $\alpha_0 = 1$); or $\mathbf{Z}_2$, hence of 2-rank equal to 1 (if $\alpha_0 = 2$); or of 2-rank equal to 2 (if $\alpha_0 > 2$). Since the 2-rank of a product is the sum of the 2-ranks, we compute the 2-rank of $\mathbf{Z}_n^*$ in terms of r. We conclude with the fact that $\omega(n) = r$ if n is odd and $\omega(n) = r + 1$, otherwise. □

So, since n to be generated in the `SIS` scheme is odd, we conclude the number t of elements used from $\mathbf{Z}_n^*$ to generate z' is such that $t \geq \omega(n) + s$. This is a sufficient condition for $P_{\mathsf{gen}} \geq 1 - 2^{-s}$.

Corollary 10. *For $t \geq \omega(n) + s$, $x_1, \ldots, x_t$ $\mathbf{Z}_2$-generate $\mathbf{Z}_n^*$ with a probability greater than or equal to $1 - 2^{-s}$.*

By the Ramanujan-Hardy theorem [9], the average number $\omega(m)$ of distinct prime factors of a random number m is $\ln(\ln m)$. Further, by the Erdös-Kac theorem [6] says that $\frac{\omega(m)-\ln\ln m}{\sqrt{\ln\ln m}}$ follows the standard normal distribution, for such a random number m. So, $\Pr\left[\omega(m) > \ln\ln m + \sqrt{2s.\ln 2.\ln\ln m}\right]$ is $F(-\sqrt{2s.\ln 2})$, where F is the standard normal cumulative distribution function.

We can apply the arguments above using the numbers α_i and β_i that the `Key generation` algorithm produces. Namely, first see that $\omega(n) \leq \Sigma_{i=1}^{2k}(\omega(\alpha_i)+\omega(\beta_i))$. So, we can bound the mean of the variable $\omega(n)$ with that expected value $2k \times \ln\ln \ell$. So,

$$\Pr\left[\omega(n) \geq 2k \times \ln\ln 2^{\ell} + \sqrt{2s \times \ln 2 \times 2k \times \ln\ln 2^{\ell}}\right] \leq F(-\sqrt{2s.\ln 2}).$$

Since $F(-x)$ can be approximated with $\frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}/x$, then the probability is smaller than $\frac{1}{\sqrt{4\pi s \ln 2}}2^{-s}$. Thus, all things considered, we can take:

$$t = \left\lceil 2k\ln\ln 2^{\ell} + \sqrt{2s.\ln 2.2k.\ln\ln 2^{\ell}} + s \right\rceil. \tag{1}$$

Hence, t can be taken of the order of $\sqrt{s \cdot k \cdot \log \ell} + s$. The final estimation of t asymptotically in s will be clear at the end of this section, after we see exactly how k and ℓ vary in s.

Note: As one can see, one dominant component in the asymptotic expression (1) of t is the standard deviation of the random variable characterising $\omega(n)$, i.e., in the term $\sqrt{2s.\ln 2.2k.\ln\ln 2^{\ell}}$. We ran several experiments with smaller α's and β's and we observed that in practice this standard deviation is in fact smaller than $\sqrt{2k \times \ln(\ln(2^{\ell}))}$ and even smaller than the value[2] of $\sqrt{\ln(2k\ell\ln 2)}$. So, in practice, t could potentially be taken smaller than the asymptotic approximation proven here.

4.2 The Local Parameters k and ℓ

It can be seen (as developed in Section 3.2) that in order for the **CHI** problem to be hard and, separately for key recovery attacks to be impossible, the factorisation of the modulus n, generated as in our cryptosystem, needs to be hard. More precisely, the factors α and β of n must be hard to find.

Let n be a positive integer and let its unique prime decomposition be as follows: $n = n_1 n_2 \cdots n_v$, with $n_1 \geq n_2 \geq \ldots \geq n_v$. In [13], Knuth *et al.* look at the probability that, for a random number n, the r^{th} largest of its prime factors, n_r, is smaller than n^x where $0 < x < 1$. We recall some commonplace notations describing this:

- $F_r(x) = \lim_{N\to+\infty}\frac{P_r(x,N)}{N}$, where $P_r(x,N) = \#\{1 \leq n \leq N | n_r \leq N^x\}$;
- $\psi(x,y) = P_1\left(\frac{\log y}{\log x}, x\right)$ is the de Brujin function [10] and the ratio $\psi(x,y)/x$ can be interpreted as the probability that an integer chosen at random in the interval $[1,x]$ has all its prime factors smaller than or equal to y. This function has several approximations [10,13].
- Thus, $F_1(x) = \lim_{N\to+\infty}\frac{\psi(N,N^x)}{N} = \rho(1/x)$ where ρ is Dickman's function. Since $\rho(u) \leq \frac{1}{u!}$, we use a convenient upper bound $(F_1(x))^{-1} \geq \frac{1}{x}!$. In [26], van de Lune and Wattel provide a numerical table for $\rho(u)$ when u is large.

We express our security desiderata (the hardness on n's factorisation); to do so, we will use some value $x \in (0,1)$. To ease our explanations, we start by recalling the complexities of factoring with elliptic curves, ECM and with the generalised number sieves, GNFS.

[2] This would be the standard deviation predicted by the Erdös-Kac theorem, if the latter were applicable directly to an n generated like ours.

– The complexity of factoring n with GNFS [15] is in $O(e^{\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}})$; so given the constants $c, \varepsilon \in \mathbb{R}$ we define a function $C_{GNFS}(L,c,\varepsilon) = c \times (e^{\left(\sqrt[3]{\frac{64}{9}}+\varepsilon\right)(\ln 2^L)^{\frac{1}{3}}(\ln\ln 2^L)^{\frac{2}{3}}}$.
We take for granted that $C_{GNFS}(1\,248) \approx 2^{80}$ [2]. We deduce that we can reasonably take $\varepsilon \approx 0$ and $c \approx 2^{-14}$. In what follows, $C_{GNFS}(L) = C_{GNFS}(L,c,\varepsilon)$.
– The complexity of factoring a number n with ECM [16] is in $O(e^{\sqrt{2+o(1)}\sqrt{\ln p \ln\ln p}})$, where p is the smallest factor of n and $|p| \approx \frac{\ln p}{\ln 2}$ is the bitlength of p; so we define a function $C_{ECM}(L,c',\varepsilon') = c' \times e^{\sqrt{2+\varepsilon'}\sqrt{\ln 2^L \ln\ln 2^L}}$. In what follows, $C_{ECM}(L) = C_{ECM}(L, 2^{-14}, 0)$ as in C_{GNFS}. So, even though one would find all factors of n of length smaller than $x\ell$, one would not isolate α.

Now, we impose our conditions to align the security of `SIS` to the security levels of factoring moduli in general, in public-key cryptography. First, we impose equation (2). This equation stipulates that factors of n with no divisors of size less than $x\ell$ are hard to find with ECM. There is one such factor in α_i, respectively in β_i, with probability $1 - F_1(x)$. Equation (4) ensures that we have at least k' of such factors in n and at least one in α or in β, respectively. Equation (3) means that the product of hard-to-find factors is also hard to factor with GNFS.

$$C_{ECM}(x\ell) \geq 2^s \tag{2}$$

$$C_{GNFS}(k'x\ell) \geq 2^s \tag{3}$$

$$\sum_{i=0}^{k'-1} \binom{2k}{i} F_1(x)^{2k-i}(1-F_1(x))^i + 2\sum_{i=k'}^{k} \binom{k}{i} F_1(x)^{2k-i}(1-F_1(x))^i \leq 2^{-s} \tag{4}$$

The latter condition would approximate to $\binom{2k}{k'} F_1(x)^{2k-k'+1} + 2F_1(x)^k \leq 2^{-s}$, where the second term could be neglected for $k' > k$. Practically, using `SAGE` [25], we derive the following parameters.

s	80	128	192	256	320	384	448	512
x	0.0561	0.0654	0.0653	0.0652	0.0651	0.0574	0.0585	0.0593
k	1	2	3	4	5	5	6	7
k'	2	3	4	5	6	6	7	8
ℓ	10978	16553	31080	50143	73204	117776	145499	181116
$k'x\ell$	1232	3248	8118	16347	28593	40562	59581	85921
$2k\ell$	21956	66212	186480	401144	732040	1177760	1745988	2535624
t	143	247	379	512	648	743	880	1018

We recall that $2k\ell$ is the modulus size and that $k'x\ell$ would be the modulus size for GM or RSA with equivalent security.

Interpreting the optimised parameters. For instance, with $s = 128$, the probability that α_i resp. β_i is $2^{x\ell}$-smooth is $F_1(x) \approx 2^{-65.3}$ and the probability that we do not have at least 3 non-smooth factors out of $2k = 4$ is $2^{-128.0}$. So, we could count on at least 3 factors with no prime divisor of length lower than $x\ell$, i.e., on a hard-to-factor integer of $3x\ell$ bits.

On choosing k. Asymptotically, we can take $x \sim \frac{\log\log s}{\log s}$, $k \sim \frac{s}{\log s}$, $k' = uk$ given a constant u, and $\ell \sim s^2$. Clearly, (2)-(3) are satisfied. We have $\binom{2k}{k'} \leq 2^{2k}$. By using $F_1(x)^{-1} \geq \frac{1}{x}!$, we can show that $F_1(x)^{-1} \geq s^{O(1)}$, so $\binom{2k}{k'} F_1(x)^{2k-k'+1} \leq 2^{-s}$ by tuning u appropriately. This makes sure that (4) is satisfied. So, we have $2k\ell \in O\left(s^3(\log s)^{-1}\right)$. Since we already showed that $t \in O(\sqrt{s.k.\log\ell} + s)$,

obtain $t \in O(s)$. We recall that (according to the note on page 9) experiments indicate that this could be a pessimistic approximation of t and suggest that, in practice, t could be taken smaller.

In contrast, $k = 1$ and $k' = 2$ leads us to $x \in O\left(\frac{\log s}{s}\right)$ then to $2k\ell = O\left(s^4(\log s)^{-3}\right)$ which is asymptotically larger than before. So, there is an advantage in taking k larger than 1 (which was maybe not intuitive to begin with).

5 Complexity of the Scheme

5.1 Asymptotic Complexity

In the modulus-generation phase, $2k$ integers of size ℓ are randomly picked and $2k-1$ multiplications are performed, in order to obtain the value n. Using a divide and conquer strategy, these operations are performed in $O(2k\ell + \Sigma_i 2^i C_{mul}(alg, \frac{k\ell}{2^i}))$, $i = 0, 1, \ldots, \log_2 2k-1$, where $C_{mul}(alg, \ell)$ denotes the complexity of multiplying numbers of ℓ bits using a particular algorithm, i.e., $C_{mul}(\text{schoolbook}, \ell) = O(\ell^2)$, $C_{mul}(\text{Karatsuba}, \ell) = O(\ell^{\log_2 3})$, $C_{mul}(\text{FFT-optimised}, \ell) = O(\ell \log \ell)$. In the next, "*sch.*" denotes the schoolbook multiplication and "FFT" denotes the FFT-based multiplication algorithm [23].

Thus, the complexity of modulus generation in SIS, $O(\text{SIS-Gen})$, is

$$O(2k\ell + \Sigma_i 2^i C_{mul}(alg, \frac{k\ell}{2^i})) = \begin{cases} O(2k\ell + \Sigma_{i=0}^{\log_2 2k-1} \frac{(k\ell)^2}{2^i}) = O((k\ell)^2), & \text{if alg is sch.} \\ O(2k\ell + \Sigma_{i=0}^{\log_2 2k-1} 2^i \frac{k\ell}{2^i} \log \frac{k\ell}{2^i}) = O(k\ell \log k\ell), & \text{if alg is FFT.} \end{cases}$$

As we will see, this is dominated by what remains in the key generation.

Indeed, the choices for the values x_i from $\mathbf{Z}_n^*$ and the computation of the values y_i are done in $O(2tk\ell + tC_{Jac}(alg, 2k\ell))$, where $C_{Jac}(alg, x)$ denotes the complexity to calculate the Jacobi symbol on an input of two x-bit integers, using the algorithm alg for the multiplication needed inside the calculation of the symbol. We know that $C_{Jac}(alg, L) = C_{mul}(alg, L) \log L$, for L being a size of the inner modulus. Thus, this complexity amounts to

$$O(2tk\ell + tC_{Jac}(alg, 2k\ell)) = O(2tk\ell + tC_{mul}(alg, 2k\ell) \log k\ell) = \begin{cases} O(t \times (k\ell)^2 \log k\ell), & \text{if alg is sch.} \\ O(t \times (k\ell)(\log k\ell)^2), & \text{if alg is FFT.} \end{cases}$$

For encryption, one picks t bits and performs at most $t+1$ multiplications to compute z, which takes $O(tC_{mul}(alg, 2k\ell))$. The value P is then computed within a complexity of order $O(t)$ as each y_i value is from $\{+1, -1\}$ and each value b_i is from $\{0, 1\}$. Similarly to the above, this gives a total complexity of SIS of the order of $\begin{cases} O(t \times (k\ell)^2), & \text{if alg is sch.} \\ O(t \times (k\ell) \log k\ell), & \text{if alg is FFT.} \end{cases}$

For decryption, we need to compute one Jacobi symbol, thus spending $C_{Jac}(alg, k\ell)$ on the procedure. This implies a complexity of $\begin{cases} O((k\ell)^2 \log k\ell), & \text{if alg is sch.} \\ O((k\ell)(\log k\ell)^2), & \text{if alg is FFT.} \end{cases}$

By SIS-s, we denote the SIS cryptosystem over the domain $\{0,1\}^s$, in which –of course– the encryption and decryption have an overhead factor of s. We do consider s-bit messages since, while having a security 2^s, one would be interested in encrypting a symmetric key of s bits to use a consistent security level in hybrid encryption. Once again, we recall that (according to the note on page 9) actual complexities may be lowered if we found tighter approximation of t.

We would now like to compare our complexity with the complexity of the GM scheme that we extend herein, as well as to that of other public-key cryptosystems based on primality-testing, e.g., RSA. Given the existent optimised implementations of RSA (based mainly on optimised multiplications),

we will consider the most expensive multiplication algorithm, e.g., schoolbook multiplication, as well as the cheapest multiplication one, e.g., FFT-based. We extend this reasoning for the case where we consider the GM cryptosystem too. We recall that the bottleneck of RSA/GM, if the key-generation is considered, is indeed the prime-generation.

An additional fact to consider in this comparison is that –in general– the key-generation in public-key cryptosystem is a one-time process, i.e., one generates the keys once and encrypts/decrypts several times. An exception to this case is, as we mentioned in the introduction, the context of secure delegation of linear algebra computation, where some client outsources e.g, the solving a linear system, to a remote worker; in these cases, the secure delegation protocols require re-generation of each run of the secret/public keys. Outside this setting, it is compelling to consider separately the case of the complexity of RSA without the key generation and draw a corresponding comparison with the complexity of th system herein, i.e., to compare the complexity of solely the encryption-decryption part of RSA with the SIS cryptosystems.

Another thing to bare in mind in this comparative study is that the SIS cryptosystem in its presented form encrypts a single bit. The GM system has also a "bit-by-bit" fashion. So, if we were to compare SIS asymptotically with the RSA that encrypts s bits at once, then we ought to consider s encryptions of the SIS cryptosystem.

For this comparison, we take the asymptotic values $k\ell \in O\left(s^3(\log s)^{-1}\right)$ and $t \in O\left(s^{\frac{3}{2}}(\log s)^{-\frac{1}{2}}\right)$ that we obtained. We consider the GM cryptosystem first. By looking carefully at the complexity of the GM cryptosystem (see Section B.1), one can conclude with Table 1.

Table 1: Asymptotic Complexities in Security Parameter s for GM vs. SIS

		key-generation	encryption	decryption
schoolbook multiplication	GM	$O(s^{12}(\log s)^{-8})$	$O(s^6(\log s)^{-4})$	$O(s^6(\log s)^{-5})$
	SIS	$O(s^7(\log s)^{-1})$	$O(s^7(\log s)^{-2})$	$O(s^6)$
FFT-based multiplication	GM	$O(s^9(\log s)^{-5})$	$O(s^3(\log s)^{-1})$	$O(s^3)$
	SIS	$O(s^4(\log s))$	$O(s^4)$	$O(s^3 \log s)$

We consider now the RSA cryptosystem. To assess the asymptotic complexity of RSA, we make the following distinction. On the one hand, one can assume a random full-size e. Let us denote this as RSA. On the other hand, one can take the public exponent of RSA as a constant, e.g., $e = 2^{16}+1$, in which case we will refer to using $\text{RSA}_{e\text{ cte.}}$. In the latter case, in the encryption, there are $O(1)$ multiplications.

By looking carefully at the complexity of the RSA cryptosystem (see Section B.2), we can wrap the complexity comparison between RSA and SIS-s in Table 2.

5.2 Experimental Results

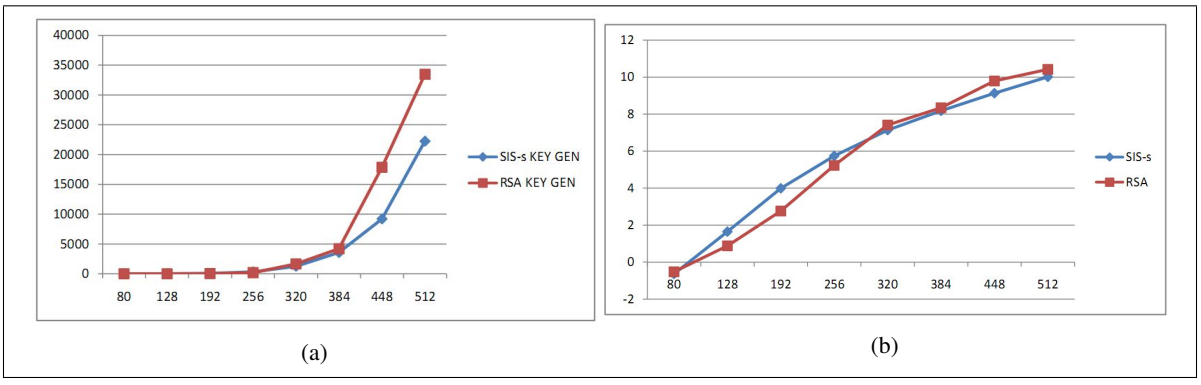
To assess the correctness of our analysis, we implemented and compared the running time of RSA and SIS-s. The experimental environment consists of a Linux kernel 3.2.0-31 that runs on a Intel Xeon 3.33Ghz CPU. The implementation was done in C and for our large numbers we use the GMP library [7]. The implementations of both SIS-s and RSA were tested for the same security parameters illustrated in the table from page 10, namely s varies from 80 to 512.

Table 2: Asymptotic Complexities in Security Parameter s for $\texttt{RSA}_{e\text{ cte.}}$ vs. SIS

		key-generation	encryption	decryption
schoolbook multiplication	$\texttt{RSA}_{e\text{ cte.}}$	$O(s^{12}(\log s)^{-8})$	$O(s^{6}(\log s)^{-4})$	$O(s^{9}(\log s)^{-6})$
	RSA	$O(s^{12}(\log s)^{-8})$	$O(s^{9}(\log s)^{-6})$	$O(s^{9}(\log s)^{-6})$
	SIS-s	$O(s^{7}(\log s)^{-1})$	$O(s^{8}(\log s)^{-2})$	$O(s^{7})$
FFT-based multiplication	$\texttt{RSA}_{e\text{ cte.}}$	$O(s^{9}(\log s)^{-5})$	$O(s^{3}(\log s)^{-1})$	$O(s^{6}(\log s)^{-3})$
	RSA	$O(s^{9}(\log s)^{-5})$	$O(s^{6}(\log s)^{-3})$	$O(s^{6}(\log s)^{-3})$
	SIS-s	$O(s^{4}(\log s))$	$O(s^{5})$	$O(s^{4}\log s)$

SIS-s implementation. Our implementation verifies the asymptotic complexities we provide in Table 2. In practice, if we compute the slope of the regression line for the logarithmic running time of key generation, encryption and decryption against $\log s$, we get 5.7 for the key generation, 6.2 for the encryption and 6.0 for the decryption algorithm. Indeed these values are slightly smaller than the ones in Table 2 as GMP has efficient implementations of its operations, e.g., multiplication or computation of the Jacobi symbol.

Comparing RSA and SIS-s. Besides verifying that the asymptotic complexity of SIS-s is valid, we are also interested in comparing the running time of RSA and SIS-s for the generation of the key. As aforementioned, this represents the bottleneck for RSA and our scheme tries to offer a speed up for those scenarios, like secure delegation of computation, where the key generation is not an one-off process.



(a) (b)

Figure 1: Key generation SIS-s vs. RSA

The above figures illustrate both the running time (Figure 1a) and the logarithmic running time (Figure 1b) of the key generation for RSA and SIS-s, where the security parameter s takes values between 80 and 512. The running time is measured in seconds. For small values of s, the generation of primes for RSA is faster than our primeless method. But one may notice that once we increase the value of the security parameter the two plots intersect at around $s = 300$ and clearly the key generation of our SIS-s becomes faster than the one of RSA. This behaviour reinforces our asymptotic study.

We conclude that, by comparison with GM and RSA, SIS exhibits improved asymptotic complexities for all procedures, apart from encryption. The result is sustained also by our experimental

results. This would solve the bottlenecks of the execution of secure delegation of computation and/or EKE password-based key exchange [4], i.e., of all settings where the key-generation is not an one-off process.

6 Conclusions

In this paper, using the same, main idea of relying on hard characters of order 2, we have extended the Goldwasser-Micali cryptosystem in a way that bypasses completely the use/generation of prime numbers. In doing so, the resulting scheme has a complexity asymptotically smaller than the one of standard public-key cryptosystems. This would yield a considerable speed-up to secure delegation protocols [19] that use homomorphic encryption schemes, GM included, in a way where the key-generation is repeated at every run of the protocol.

It is possible to improve the efficiency of our cryptosystem by using characters of higher order. For instance for characters of order 4, with the quartic residue symbol, the two participants can encrypt two bits instead of one. In this scenario, participant A is choosing β_i and α_i to be Gaussian integers and computes n as $\prod_i \alpha_i \cdot \bar{\alpha}_i \cdot \beta_i \cdot \bar{\beta}_i$, where $\bar{\gamma}_i$ is simply the complex-conjugate of a Gaussian integer γ_i. The B participant is now choosing the values b_i from $\{0,1,2,3\}$. The correctness and the security proof of this optimised scheme are maintained (i.e., similar to the case of our cryptosystem, with reductions to the **MOVA**$^{\mathbf{4}}$ and a different proof on the distribution spawned by the values z generated in such a scheme). However, the complexity of a scheme thus-wise optimised is higher than the one herein presented.

Still, using only characters of order 2, our cryptosystem can be extended so that it directly encrypts more than 1 bit by using algebraic codes. This extension is not the subject of this paper, being left for future work.

As future work, we would also like to study further the optimisation problem on our parameters, implied by our computational hardness constraints. It would be interesting to study from closer the standard deviation of $\omega(n)$, to find a tighter asymptotic approximation of t.

References

1. Institute of Electrical and Electronics Engineers. Ieee standard specifications for public key cryptography. IEEE 1363-2000, 2000. http://grouper.ieee.org/groups/1363/.
2. Institute of Electrical and Electronics Engineers. ECRYPT II Yearly Report on Algorithms and Keysizes. ECRYPT, 2011. http://www.ecrypt.eu.org/documents/D.SPA.17.pdf.
3. M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. *The Annals of Mathematics*, 160(2):pp. 781–793, 2004.
4. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY*, pages 72–84, 1992.
5. H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.
6. P. Erdös and M. Kac. The gaussian law of errors in the theory of additive number theoretic functions. *American Journal of Mathematics*, 62(1):738–742, 1940.
7. *The GNU Multiple Precision Arithmetic Library*. http://gmplib.org.
8. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
9. G. Hardy and S. Ramanujan. The normal number of prime factors of a number *n*. *Quart. J. Math.*
10. A. Hildebrand and G. Tenenbaum. *Integers without large prime factors*. Prépublications de l'Institut Elie Cartan. Dép. de Math., Univ. de Nancy I, 1991.
11. K. Ireland and M. Rosen. A classical introduction to modern number theory. 1990.
12. A. Karatsuba and Yu. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*.
13. D. E. Knuth and L. T. Pardo. Analysis of a simple factorization algorithm. *Theoretical Computer Science*, 3(3):321348, 1976.

14. A. K. Lenstra and H. W. Lenstra, Jr. *Algorithms in number theory*, pages 673–715. MIT Press, Cambridge, MA, USA, 1990.
15. A. K. Lenstra and H. W. Lenstra, Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.
16. H. W. Lenstra, Jr. Factoring integers with elliptic curves. *Ann. of Math. (2)*, 126(3):649–673, 1987.
17. Jr. Lenstra and Hendrik W. Primality testing with gaussian periods. In *Proceedings of the 22nd Conference Kanpur on Foundations of Software Technology and Theoretical Computer Science*, FST TCS '02, pages 1–. Springer-Verlag, 2002.
18. Robert J. Mceliece. A public-key cryptosystem based on algebraic coding theory. Technical report, Jet Propulsion Lab Deep Space Network Progress report, 1978.
19. P. Mohassel. Efficient and secure delegation of linear algebra. Cryptology ePrint Archive, Report 2011/605, 2011. `http://eprint.iacr.org/`.
20. J. Monnerat. *Short undeniable signatures: Design, Analysis, and Applications*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2006.
21. J. Monnerat and S. Vaudenay. Short Undeniable Signatures Based on Group Homomorphisms. *Journal of Cryptology*, 24(3):545–587, 2011.
22. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21:120–126, February 1978.
23. A. Schönhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7:281–292, 1971.
24. R. Schoof. Counting points on elliptic curves over finite fields, 1995.
25. W. A. Stein et al. *Sage Mathematics Software*. The Sage Group, 2009. `http://www.sagemath.org`.
26. J. van de Lune and E. Wattel. On the numerical solution of a differential-difference equation arising in analytic number theory. *Mathematics of Computation*, 23:417–421, 1969.

A Proofs

Theorem 3. *Given n, $x_1,\ldots,x_t,y_1,\ldots,y_t$ defining a unique χ on $\mathbf{Z}_n^*$, if $\mathcal{A}$ is a probabilistic algorithm which is polynomial in s such that*

$$\left|\Pr\left[\mathcal{A}(x)=\chi(x)\,|x\in_U \mathbf{Z}_n^*\right]-\frac{1}{2}\right|>\theta,$$

then an algorithm $\mathcal{A}'$ calling $\mathcal{A}$ a number of $\frac{1}{2}\theta^{-2}\ln\frac{2}{\varepsilon}$ times can be defined such that

$$\Pr\left[\mathcal{A}'(x)=\chi(x)\,|x\in_U \mathbf{Z}_n^*\right]\geq 1-\varepsilon,$$

with $\theta>0$ and $\varepsilon>0$.

Proof. Let $\mathcal{A}$ be a probabilistic algorithm which is polynomial in s. Let

$$p=\Pr\left[\mathcal{A}(x)=\chi(x)|x\in_U \mathbf{Z}_n^*\right]$$

We define the following algorithm $\mathcal{A}'(x)$:

initialize $c_1\leftarrow 0$ and $c_2\leftarrow 0$
for $i=1$ to k **do**
  pick some random bits $b_1,\ldots,b_t$ and $r\in_U \mathbf{Z}_n^*$
  set $x'\leftarrow xx_1^{b_1}\cdots x_t^{b_t}r^2 \bmod n$
  $c_1\leftarrow c_1+\mathcal{A}(x')y_1^{b_1}\cdots y_t^{b_t}$
  pick some random bits $b_1,\ldots,b_t$ and $r\in_U \mathbf{Z}_n^*$
  set $x'\leftarrow x_1^{b_1}\cdots x_t^{b_t}r^2 \bmod n$
  $c_2\leftarrow c_2+\mathcal{A}(x')y_1^{b_1}\cdots y_t^{b_t}$
end for
output the sign of c_1c_2

Since $p \neq \frac{1}{2}$, we observe that x' at step 7 is uniformly distributed in $\mathbf{Z}_n^*$ and such that $\chi(x') = y_1^{b_1} \cdots y_t^{b_t}$. For this, see Lemma 11 given below.

So, c_2 is incremented with probability p and decremented with probability $1-p$. Due to the Chernoff bound [5], the final value of c_2 has the sign of $p - \frac{1}{2}$ with probability at least $1 - e^{-2k(p-\frac{1}{2})^2}$.

Similarly, c_1 multiplied by the sign of $p - \frac{1}{2}$ will have the sign of $\chi(x)$ with probability at least $1 - e^{-2k(p-\frac{1}{2})^2}$.

So, the algorithm produces the correct value of $\chi(x)$ with probability at least $1 - 2e^{-2k(p-\frac{1}{2})^2}$.

By taking $k = \frac{1}{2}\theta^{-2} \ln \frac{2}{\varepsilon}$ and since $|p - \frac{1}{2}| > \theta$, we attain the required probability of "success" for the algorithm $\mathcal{A}'$ thus-wise constructed, where ε is the "error" in the text of the theorem.

□

Lemma 11 (Lemma 4.16 in [21]). *Let G and H be two finite Abelian groups. We denote by d the order of H, where the group operation is denoted additively. Let $x_1, x_2, \ldots, x_r \in G$. If $x_1, \ldots, x_t$ H-generate G then by picking $r \in_U dG$ and $b_1, \ldots, b_t \in_U \{0, \ldots, d-1\}$ then $dr + b_1 x_1 + \cdots + b_t x_t$ is uniformly distributed.*

Theorem 4. *If the QR problem is hard relative to Gen_{GM}, then **CHI** problem is hard relative to Gen_{CHI}.*

Proof. Let $\mathcal{A}$ be an adversary against CHI. Let $\mathsf{Gen}_{QR}(1^s) \to (n)$. We pick one of the two hard characters χ at random. Let

$$p_n = \Pr[\mathcal{A}(x, n, x_1, x_2, \chi(x_1), \chi(x_2)) = \chi(x) | n, \mathsf{span}_2(x_1, x_2) = \mathbf{Z}_n^*]$$

over $x, x_1, x_2 \in_U \mathbf{Z}_n^*$ and χ. Due to the definition of Gen_{CHI}, what we have to prove is that $E(p_n)$ is negligible when the QR problem is hard relative to Gen_{QR}.

We construct an adversary $\mathcal{B}(u, n)$ against QR as follows. By definition, we have $(u/n) = +1$. Then, we pick $v \in_U \mathbf{Z}_n^*$ until $(v/n) = -1$ and $\sigma \in_U \{-1, +1\}$. Let χ be the only hard character of $\mathbf{Z}_n^*$ of order 2 such that $\chi(v) = \sigma$. Clearly, χ is a uniformly distributed hard character. Then, we select bits a, b, c, d until $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ is odd. Finally, $r, r' \in_U \mathbf{Z}_n^*$. We define $x_1 = u^a v^b r^2 \bmod n$ and $x_2 = u^c v^d (r')^2 \bmod n$.

The residue u is quadratic if and only if $\chi(u) = +1$. When it is not, then $\mathsf{span}_2(u, v) = \mathbf{Z}_n^*$. In that case, (x_1, x_2) is randomly distributed over the pairs such that $\mathsf{span}_2(x_1, x_2) = \mathbf{Z}_n^*$. Still in the case that $\chi(u) = -1$, we note that $\chi(x_1) = (-1)^a \sigma^b$, $\chi(x_2) = (-1)^c \sigma^d$. Let $x = u^\alpha v^\beta (r'')^2 \bmod n$ with $\alpha, \beta \in_U \{0, 1\}$ and $r'' \in_U \mathbf{Z}_n^*$. Clearly, x is uniformly distributed in $\mathbf{Z}_n^*$ and $\chi(x) = (-1)^\alpha \sigma^\beta$. Thanks to the good distributions, we have

$$\Pr[\mathcal{A}(x, n, x_1, x_2, (-1)^a \sigma^b, (-1)^c \sigma^d) = (-1)^\alpha \sigma^\beta | n, \chi(u) = -1] = p_n$$

In the case $\chi(u) = +1$, the inputs to $\mathcal{A}$ are independent from α. So, the above probability becomes $\frac{1}{2}$.

We define

$$\mathcal{B}(u, n) = 1_{\mathcal{A}(x, n, x_1, x_2, (-1)^a \sigma^b, (-1)^c \sigma^d) \neq (-1)^\alpha \sigma^\beta}$$

Clearly, for $(u/n) = -1$, we have $\Pr[\mathcal{B}(u, n) = 0 | n] = p_n$. For $(u/n) = -1$, we have $\Pr[\mathcal{B}(u, n) = 1 | n] = \frac{1}{2}$. So,

$$\Pr[\mathcal{B}(u, n) = 1_{u \in QR_n}] = \frac{1}{4} + \frac{E(p_n)}{2}$$

Assuming that the QR problem is hard relative to Gen_{QR}, we obtain that $\left| \frac{E(p_n)}{2} - \frac{1}{4} \right|$ is negligible. Therefore, $|E(p_n) - \frac{1}{2}|$ is negligible as well. Since this holds for all $\mathcal{A}$, we deduce that the CHI problem is hard relative to Gen_{CHI}.

□

B Detailed Complexity Comparison

B.1 The Complexity of Goldwasser-Micali Cryptosystem

We start by considering the GM cryptosystem, namely the encryption/decryption of one bit under GM. We take $L = O(s^3(\log s)^{-2})$. In the key-setup stage, generating the GM-modulus N as a product of two primes p, q with $|p| = |q| = L$ takes $O(L^2 \times C_{mul}(alg, L))$ and computing the Jacobi symbol needed has a complexity $C_{Jac}(alg, 2L) = C_{mul}(alg, 2L)\log 2L$. The GM-encryption has the asymptotic complexity of $C_{mul}(alg, 2L)$ (it does two or three multiplications modulo N). The GM-decryption algorithm for one bit has the complexity of $C_{Jac}(alg, 2L) = C_{mul}(alg, 2L)\log 2L$. Thus, we have

$$O(\texttt{GM key-generation}) = \begin{cases} O(L^4) = O(s^{12}(\log s)^{-8}), & \text{with sch. multiplication} \\ O(L^3 \log L) = O(s^9(\log s)^{-5}), & \text{with FFT-based multiplication} \end{cases}$$

$$O(\texttt{GM encryption}) = \begin{cases} O(4 \times L^2) = O(s^6(\log s)^{-4}), & \text{with sch. multiplication} \\ O(2 \times L \times \log 2L) = O(s^3(\log s)^{-1}), & \text{with FFT-based multiplication} \end{cases}$$

$$O(\texttt{GM decryption}) = \begin{cases} O(4 \times L^2 \times \log 2L) = O(s^6(\log s)^{-5}), & \text{with sch. multiplication} \\ O(2 \times L \times (\log 2L)^2) = O(s^3), & \text{with FFT-based multiplication} \end{cases}$$

B.2 The Complexity of the RSA Cryptosystem

We now consider the RSA cryptosystem. We take $L = O(s^3(\log s)^{-2})$. We start with the RSA decryption. To exponentiate to the power an exponent b of size L in bits, we do $O(L \times C_{mul}(alg, L))$, which gives $O(L^3)$, if schoolbook multiplication is used and $O(L^2 \log L)$, if FFT-based multiplication is used. So,

$$O(\texttt{RSA decryption}) = \begin{cases} O(L^3) = O(s^9(\log s)^{-6}), & \text{with schoolbook multiplication} \\ O(L^2 \log L) = O(s^6(\log s)^{-3}), & \text{with FFT-based multiplication.} \end{cases}$$

We can take two views on the RSA encryption. On the one hand, one can assume that the exponentiation in the RSA encryption is *not* done in constant number of multiplications, by having chosen a constant public exponent e. Let us denote this as RSA. Thus, for a choice of the public RSA exponent e lying indeed in $\mathbb{Z}^*_{\phi(n)}$, one get $O(\texttt{RSA encryption}) = O(\texttt{RSA decryption})$. On the other hand, one can take the public exponent of RSA as a constant, in which case we will refer to using $\texttt{RSA}_{e\text{ cte.}}$. Then,

$$O(\texttt{RSA}_{e\text{ cte.}}\texttt{encryption}) = C_{mul}(alg, L) = \begin{cases} O(L^2) = O(s^6(\log s)^{-4}), & \text{with schoolbook multiplication} \\ O(L \log L) = O(s^3(\log s)^{-1}), & \text{with FFT-based multiplication.} \end{cases}$$

The RSA key-generation runs in $O(L \times 2 \times \frac{L}{2} \times C_{mul}(alg, L))$. So, $O(\texttt{RSA key-generation}) =$
$\begin{cases} O(s^{12}(\log s)^{-8}), & \text{with schoolbook multiplication} \\ O(s^9(\log s)^{-5}), & \text{with FFT-based multiplication} \end{cases}$

C Other Cryptosystems, Not Essentially Prime-Based

If working with elliptic curves, the difficulty of the discrete logarithm problem for the group $E(\mathbb{F}_q)$ of elliptic curves over a finite field $\mathbb{F}_q$, where $q = p^k$ and p is a prime, is judged by the number of points on the generated curves; algorithms [24] for point-counting run at in $O(\frac{N^2 C_{mul}(alg, N^2)}{\log N})$ where $N = \log q$, $C_{mul}(alg, \ell)$ denotes the complexity of multiplying numbers of ℓ bits using a particular algorithm. This complexity can be approximated to $O(N^4 + o(1))$, which asymptotically in a security parameter s can be approximated to $O(s^6)$. This method yields smaller key-lengths than in the case of RSA, the bottleneck of the time-complexity is reduced, yet the space complexity needed is of the space complexity is $O(N^4 \log N)$, hence still of around $O(s^6)$.

Less used asymmetric cryptosystems, e.g., the McEliece cryptosystem [18] based on algebraic codes, are faster than traditional lines and have a better security increase with the growth of the key-length. Nevertheless, the McEliece private and public keys are large matrices of the order of $5 * 10^5$ bits. Numerically, this size proves to be much larger than that the key in the cryptosystem herein proposed. McEliece is interestingly still *asymptotically* faster than standard public-key cryptography lines.