CCA1-secure somewhat homomorphic encryption

Eduardo Morais[1], **Diego F. Aranha**[1], **Ricardo Dahab**[1]

[1]Institute of Computing – University of Campinas (Unicamp)

{dfaranha, rdahab}@ic.unicamp.br, emorais@lasca.ic.unicamp.br

Abstract. *This paper proposes the combination of homomorphic encryption and verifiable computation to avoid key recovery attacks and achieve CCA1-secure constructions of Somewhat Homomorphic Encryption (SHE) schemes described in the literature. We also provide concrete parameters, based on the best-attack analysis, concluding that the approximate greatest common divisor (AGCD) [van Dijk et al. 2010] family of SHE schemes may be the best implementation choice under certain circumstances.*

1. Introduction

Homomorphic encryption has been a topic of great interest for the Cryptology community over the last few years, since Craig Gentry's breakthrough in 2009 [Gentry 2009]. Ciphertexts generated by a homomorphic encryption scheme can be algebraically manipulated, i. e. it is possible to add and multiply ciphertexts preversing the same operations over the corresponding plaintexts. A *Fully Homomorphic Encryption* (FHE) is one that allows us to run arbitrary algorithms over encrypted data. In other words, given an arbitrary algorithm $\mathcal{A}$, it is possible to find an equivalent algorithm $\mathcal{A}'$ such that, if we consider the input parameters of $\mathcal{A}'$, say $(c_1, \ldots, c_t)$, we have that they correspond to the componentwise encryption of the input parameters of $\mathcal{A}$, say $(m_1, \ldots, m_t)$. Furthermore, we have that the output of $\mathcal{A}'$ is an encryption of the output of $\mathcal{A}$ when executed using input arguments as described above. Although Fully Homomorphic Encryption (FHE) is yet to become practical, Somewhat Homomorphic Encryption (SHE) schemes can be used to construct practical applications. An important drawback, however, is the fact that all but one of the SHE schemes described in the literature are susceptible to *key recovery attacks*, a concrete threat in many scenarios. In this work, we investigate how verifiable computation (see Section 1.2) can be combined with homomorphic encryption in order to avoid key recovery attacks. Indeed, we show that it is possible to achieve CCA1-security for the homomorphic evaluation of quadratic multivariate polynomials.

1.1. Main contributions

During the PhD program I have participated in a research project about efficient implementation of elliptic curve protocols for Android architecture, giving rise to two publications [Braga and Morais 2014, Braga et al. 2015]. Since this research area is not related to homomorphic encryption, we are not going to explore it further here.

We have contributed in a book chapter, that corresponds to an introduction to Lattice-based Cryptography [Barreto et al. 2014]. This work derived from a short course in 2013 [Barreto et al. 2013] and gave rise to a technical report [Morais et al. 2016b]. We have also contributed with an introduction to homomorphic encryption [Dahab and Morais 2012], which is the text of a short course that gave rise to another technical report [Morais et al. 2016a].

One of the main results of my thesis is a key-recovery attack for the NTRU-based family of SHE schemes [Dahab et al. 2015]. We then contributed constructively to this line of research, showing how verifiable computation can be used to construct CCA1-secure SHE schemes [Morais et al.], for the case of quadratic multivariate polynomial functions.

1.2. Homomorphic verifiable computation

Although homomorphic encryption is a very flexible cryptographic primitive, when applied to the cloud computing scenario it lacks an important property: the ability to verify if a given homomorphic computation corresponds to what the client desired. A verifiable computation scheme could solve this problem, provided two requirements are met. First, the cloud must not spend much more time to perform the verifiable computation when compared to the non-verifiable solution. Second, the client must be able to verify the result faster than the time it takes to perform the entire computation by himself. There are proposals [Gennaro et al. 2010, Chung et al. 2010] that use homomorphic encryption to construct a verifiable scheme, because it is possible to offer input and output privacy, since both are encrypted. However, the underlying security model does not allow verification queries. Recently, Fiore, Gennaro and Pastro [Fiore et al. 2014] proposed a new construction that does allow verification queries, improving on the security model. They showed how to solve practical problems, such as computing quadratic multivariate polynomials over encrypted data, which can be used to homomorphically compute statistical functions. We remark that this application requires only one level of multiplications, which is an important characteristic to be considered in order to calculate the parameters of the underlying SHE scheme.

Definition 1.1. *A verifiable computation scheme $\mathcal{VC}$ is defined by the algorithms* $(\text{KEYGEN}, \text{PROBGEN}, \text{COMPUTE}, \text{VERIFY})$*, as follows:*

Key generation. *Algorithm* $\text{KEYGEN}(1^\lambda, f)$ *generates secret key* sk *and evaluation key* esk*.*

Problem generation. *Using secret key* sk*, algorithm* PROBGEN *receives as input ciphertext* c_i *and computes the corresponding authentication tag* σ_i*, such that* $\sigma_i = \text{AUTH}_{vk}(c_i, (\cdot, i))$*.*

Verification. *Given the secret* sk*, tag* σ *and ciphertext* c*, we have that* $\text{VERIFY}_{sk}(\sigma, c)$ *returns* 1 *if* $c = f([c_i])$ *and* $\sigma = \text{AUTH}_{vk}(c, (\cdot, i))$*. Otherwise, it returns* 0*.*

Evaluation. *Given* $\sigma_1, \ldots, \sigma_t$ *and the description of function* f*, algorithm* $\text{COMPUTE}_{esk}([\sigma_i], \Delta, f)$ *returns the authentication tag* σ *that corresponds to the ciphertext* $c = \text{EVAL}_{edk}([c_i], f)$ *obtained by running the* EVAL *algorithm from the underlying homomorphic encryption scheme. We say that the* $\mathcal{VC}$ *scheme is* correct *if* $\text{VERIFY}_{sk}(\sigma, c)$ *outputs* 1*.*

1.3. Homomorphic encryption

Informally, homomorphic encryption provides the possibility of having a pair of encryption and decryption functions, ENC, DEC, that allows the computation of a function f on an encrypted text c, such that $\text{DEC}(f(c)) = f(m)$, where $c = \text{ENC}(m)$. That is, allowing

functions to be computed on encrypted texts without the need for decrypting them first. The following two definitions formalize this notion.

Definition 1.2. *Correctness. A scheme* $\mathcal{E}(\text{KEYGEN}, \text{DEC}, \text{ENC}, \text{EVAL})$ *is* correct *if, for a determined circuit* $\mathbf{C}$ *and every key pair* (sk, pk)*, where* sk *is the private key and* pk *is the public key generated by* KEYGEN*, any message tuple* $\overline{m} = \langle m_1, \ldots, m_t \rangle$ *and corresponding ciphertexts* $\overline{c} = \langle c_1, \ldots, c_t \rangle$*, that is,* $c_i = \text{ENC}_{pk}(m_i)$ *for* $1 \leq i \leq t$*, then we have that*

$$\text{DEC}_{sk}(\text{EVAL}_{pk}(\mathbf{C}, \overline{c})) = \mathbf{C}(\overline{m}).$$

Furthermore, algorithms KEYGEN*,* DEC*,* ENC *and* EVAL *must have polynomial complexity.*

Definition 1.3. *Fully Homomorphic Encryption. A scheme* $\mathcal{E}$ *is* correct for a class $\mathbf{S}_\mathbf{C}$ *of circuits, if it is correct for each* $\mathbf{C} \in \mathbf{S}_\mathbf{C}$*. Moreover,* $\mathcal{E}$ *is denominated* fully homomorphic *if it is correct for every algebraic circuit, or, equivalently, if it is correct for every Boolean circuit.*

SHE schemes correspond to homomorphic encryption schemes that are correct for circuits whose multiplicative depth is limited by a certain upper bound, denoted by ℓ.

1.4. Security model

We say that a cryptosystem is secure against *chosen ciphertext attacks* (CCA2) if there is no polynomial time adversary that can win the following game with non-negligible probability.

Setup. The challenger obtains $(\mathtt{sk}, \mathtt{pk}) = \text{KEYGEN}(\lambda)$, where λ is a security parameter, and sends $\mathtt{pk}$ to the adversary $\mathcal{A}$.

Queries. $\mathcal{A}$ sends ciphertexts to the challenger, before or after the challenge, who returns the corresponding plaintexts.

Challenge. The adversary randomly generates two plaintexts $m_0, m_1 \in \mathcal{M}$ and sends to the challenger, who then randomly chooses a bit $b \in \{0, 1\}$ and computes the ciphertext $c = \text{ENC}_{\mathtt{pk}}(m_b)$. The challenger sends c to $\mathcal{A}$.

Answer. $\mathcal{A}$ sends a bit b' to the challenger and wins the game if $b' = b$.

If we allow queries only before the challenge, we say that the cryptosystem is secure against CCA1 adversaries (lunchtime attacks). Queries can be interpreted as accesses to a *decryption oracle*. If, instead, we only allow access to an *encryption oracle*, namely the adversary can choose any message to be encrypted under the same key pair, then we say that the cryptosystem is secure against *chosen plaintext attacks* (CPA).

In homomorphic encryption, it is impossible to achieve CCA2 security, because the adversary can simply add to the encrypted message some encryption of zero, which can be obtained by querying the encryption oracle, and send it back to the decryption oracle. Many FHE schemes have as public value an encryption of the private key bits, which can be sent to the decryption oracle before the challenge, making such schemes insecure

against CCA1 adversaries. Indeed, a *key recovery* attack is stronger than a CCA1 attack; also, Loftus et al [Loftus et al. 2011] showed that Gentry's construction over ideal lattices is vulnerable to key recovery attacks and presented the only somewhat homomorphic encryption scheme that is known to be CCA1-secure.

In 2015, Dahab, Galbraith and Morais showed that the NTRU-based family of SHE schemes is vulnerable to key recovery attacks [Dahab et al. 2015]. Hence, except for Loftus et al's [Loftus et al. 2011] scheme, no other known SHE proposal achieves CCA1 security.

2. The scheme

The secret key encryption scheme that can homomorphically compute the quadratic multivariate polynomial f over encrypted input is defined as follows:

Key generation. Given the description of the function f, let $\mathcal{E}_{\text{CPA}}$ be a CPA-secure secret-key homomorphic encryption scheme and let $\mathcal{VC}$ be a private and adaptively secure verifiable computation scheme as previously defined and containing the inherent message authentication algorithm HomMAC. We compute $(\texttt{dk}) = \mathcal{E}_{\text{CPA}}.\textsc{KeyGen}(1^\lambda, f)$ and $(\texttt{sk}, \texttt{esk}) = \mathcal{VC}.\textsc{KeyGen}(1^\lambda, f)$. The secret key is given by $(\texttt{dk}, \texttt{sk})$ and the evaluation key is given by $(\texttt{edk}, \texttt{esk})$.

Encryption. For $m \in \mathcal{M}$ and multi-label (Δ, i), if the multi-label was not previously used, compute $c = \mathcal{E}_{\text{CPA}}.\textsc{Enc}_{\texttt{dk}}(m)$, compute

$$\sigma = \text{HomMAC}.\textsc{Auth}_{\text{vk}}(c, \Delta, i)$$

and output $(c, \sigma, \Delta, i, f_{\text{ID}})$.

Decryption. For $(c, \sigma, \Delta, i, f^\star) \in \mathcal{C}$, when $f^\star$ is the identity function f_{ID}, if $\text{HomMAC}.\textsc{Ver}_{\text{vk}}(\sigma, c, \Delta, i)$ rejects then return $\perp$. When $f^\star = f$, if $\mathcal{VC}.\textsc{Verify}_{\texttt{sk}}(\sigma, \Delta, f)$ rejects then return $\perp$, otherwise return

$$m = \mathcal{E}_{\text{CPA}}.\textsc{Dec}_{\texttt{dk}}(c).$$

Evaluation. Given the evaluation key $\texttt{esk}$ and $([c_i], [\sigma_i], f)$, for $1 \leq i \leq t$, return (c, σ), where $(c, \sigma) = \mathcal{VC}.\textsc{Compute}_{\texttt{esk}}([c_i], [\sigma_i], f)$.

3. AGCD-based scheme

In this section we describe the construction of homomorphic encryption over the integers, which was originally proposed by Dijk, Gentry, Halevi and Vaikuntanathan [van Dijk et al. 2010], and was improved many times afterwards [Coron et al. 2014, Coron et al. 2012]. Among these improvements, we focus on batch computation by extending the original idea to apply the Chinese Remainder Theorem [Cheon et al. 2013]. The secret-key somewhat homomorphic cryptosystem is defined as follows:

Definition 1. *Let λ be the security parameter and consider the parameters ρ, η, γ as functions of λ. The algorithm* KeyGen *randomly generates the secret key dk as an odd integer p with bit-length η. To encrypt a message $m \in \mathbb{Z}_Q$, the algorithm* Enc *randomly chooses the integers r with ρ bits and q with γ/η bits and computes the ciphertext:*

$$c = qp + Qr + m.$$

The decryption algorithm computes $m = \text{DEC}_{dk}(c) = [c]_p \pmod{Q}$. *It is easy to see that the encryption is a ring homomorphism.*

Definition 2. *Consider the following distribution*

$$\mathcal{D}_{\gamma,\rho}(p) = \{qp + r \mid \quad q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p), \\ r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)\}.$$

Given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$*, finding* p *is a problem called* approximate greatest common divisor (AGCD).

The AGCD problem was studied by Howgrave-Graham in the context of cryptanalysis [Howgrave-Graham 2001]. To obtain a secure cryptosystem, parameters ρ, η, γ must be chosen to resist against attacks described in the literature [Lepoint 2014].

Definition 3. *The symmetric scheme described above can be adapted to allow batch operations [Cheon et al. 2013] as follows. Let* ℓ *be the number of slots in the the plaintext space* $\mathcal{M}$ *and* w *for the bit-length of each slot. We assume that* (ℓ, w) *are fixed and public values. Also, we use secondary security parameters* (ρ, η, γ) *to describe the cryptosystem. Since these parameters are functions of the primary security parameter* λ*, and are intimately connected to the complexity of the best-known attacks to the AGCD problem, we postpone the concrete description in order to get a cleaner definition of the scheme.*

Key Generation. *The* $\text{KEYGEN}(1^\lambda, f)$ *algorithm chooses pairwise coprime integers* Q_j *with* w *bits, for* $1 \leq j \leq \ell$*, and pairwise coprime* p_j *with* η *bits, for* $0 \leq j \leq \ell$*. We have that* Q_j *represents the size of each plaintext slot, while the plaintext space is given by* $\mathcal{M} = \mathbb{Z}_{Q_1} \times \cdots \times \mathbb{Z}_{Q_\ell}$*. Note that* $\mathcal{M}$ *is isomorphic to* $\mathbb{Z}_Q$ *for* $Q = \prod_{j=1}^{\ell} Q_j$*. The algorithm computes* $p = \prod_{j=0}^{\ell} p_j$*. The ciphertext space is given by* $\mathcal{C} = \mathbb{Z}_p$*. The secret key is given by* $dk = [p_j]$ *and the evaluation key is* $edk = p$*.*

Encryption. *Given* $[m_j] \in \mathcal{M}$*, algorithm* $\text{ENC}_{dk}([m_j])$ *chooses a random integer* r_0 *in the interval* $(-p_0/2, p_0/2]$ *and the random integers* $r_1, \ldots, r_\ell$ *with* ρ *bits. The ciphertext is computed by*

$$c = \text{CRT}(r_0, [m_j + r_j Q_j]),$$

where CRT *returns the unique integer modulo* p *that is congruent to* r_0 *modulo* p_0 *and congruent to* $m_j + r_j Q_j$ *modulo* p_j*, for every* j*. Thus the output is equal to* $c = \text{ENC}_{dk}(m)$*.*

Decryption. *Given* $c \in \mathcal{C}$*, the* $\text{DEC}_{dk}(c)$ *algorithm computes*

$$m_j \equiv c \pmod{p_j} \pmod{Q_j},$$

for $1 \leq j \leq \ell$ *and outputs* $[m_j] = \text{DEC}_{dk}(c)$*.*

Evaluation. *Homomorphic operations are carried out by simply adding and multiplying integers modulo* p*.*

The construction makes use of the following parameters:

- γ is the bit-length of ciphertexts. This parameter must be large enough in order to avoid attacks against the AGCD problem, such as the ones derived from Coppersmith's method, as for example Howgrave-Graham and Cohn-Heninger attacks, the simultaneous Diophantine approximation strategy of Lagarias and Nguyen and Stern's orthogonal lattice attack. In summary, we have that these attacks lead to the condition $\gamma = \eta^2 \Omega(\lambda)$, as described in Tancrède Lepoint's PhD thesis [Lepoint 2014];

- η is the bit-length of secret key p_j. It must be large enough to accommodate the noise growth after homomorphic operations. However, it must also be quadratic in the security parameter in order to avoid the elliptic curve method (ECM) factorization attack;
- ρ is the bit-length of the noise r_j. This parameter must be chosen satisfying $\rho = \Omega(\lambda)$, such that the scheme resists attacks against the noise [Chen and Nguyen 2012].

4. Choice of parameters and conclusion

We followed the ideas presented in Tancrède Lepoint PhD thesis [Lepoint 2014] in order to choose parameters to our scheme. Firstly, we compute η to avoid the ECM factorization attack. Then we establish an upper bound for ρ and γ. Thus, we can compute the ciphertext size to resist against the orthogonal lattice attack by decreasing the value of the noise parameter ρ until it is secure against Chen and Nguyen's attack. Afterwards, since we have calculated the values of ρ, η and γ as in Table 1, we can obtain the scheme parameters ℓ and w, with which we can calculate the plaintext size and the overhead of the scheme. We implemented a routine in Sage to calculate these values and the result is shown in Table 2. To compute ℓ we had to decrease the size of the ciphertext from γ to $\gamma = \gamma' - (\ell - 1)\eta$. In [Cheon et al. 2013], the authors show that the $\text{AGCD}_{1,\gamma}$ problem, the usual approximate GCD problem, can be reduced to the $\text{AGCD}_{\ell,\gamma'}$ problem, i.e. the CRT-based construction defined in Section 3. Then we have recomputed the ciphertext size using the relation $\gamma' = 1.5\gamma$ and calculated the number of slots using $\ell = \gamma/2\eta$.

λ	ρ	η	γ (MBits)
80	96	351	1.78
112	94	475	3.27
128	92	603	5.28

Table 1. AGCD parameters

λ	γ' (MBits)	ℓ	w	m (KBits)	overhead
80	2.67	2535	67	170	15.70
112	4.90	3442	131	451	10.87
128	7.92	4378	197	862	9.18

Table 2. Low overhead configuration

If we smaller ciphertexts are desired, then we can use the relations $\gamma' = 1.1\gamma$ and $\ell = \gamma/10\eta$, resulting in a larger overhead, as shown in Table 3.

In this work we proposed the utilization of verifiable computation to mitigate the key recovery attack problem in the context of homomorphic encryption. The proposed scheme however can only evaluate quadratic multivariate polynomials. This is enough to compute some statistical functions over encrypted data under the CCA1 security model. Remarkably, regarding this scenario, our AGCD-based construction offers an interesting choice of parameters.

λ	γ' (MBits)	ℓ	w	m (KBits)	overhead
80	1.95	507	67	34	57.40
112	3.59	688	131	90	39.88
128	5.80	875	197	172	33.72

Table 3. Smaller ciphertext configuration

References

Barreto, P. S. L. M., Biasi, F. P., Dahab, R., López-Hernández, J. C., Morais, E. M., Oliveira, A. D. S., Pereira, G. C. C. F., and Ricardini, J. E. (2013). Introdução a criptografia pós-quântica. In de Castro Andrade, R. M., editor, *Minicursos do XIII Simpósio em SeguranÃ§a da Informação e de Sistemas Computacionais. 193ed.*

Barreto, P. S. L. M., Biasi, F. P., Dahab, R., López-Hernández, J. C., Morais, E. M., Oliveira, A. D. S., Pereira, G. C. C. F., and Ricardini, J. E. (2014). A panorama of post-quantum cryptography. In *Open Problems in Mathematics and Computational Science. 1ed.: Springer International Publishing*, pages 387–439.

Braga, A. M. and Morais, E. M. (2014). Implementation issues in the construction of standard and non-standard cryptography on android devices. In *Securware, The Eighth International Conference on Emerging Security Information, Systems and Technologies*, pages 144–150.

Braga, A. M., Morais, E. M., Schwab, D. C., Vannucci, A. L., and Zanco Neto, R. (2015). Integrated technologies for communication security and secure deletion on android smartphones. *ICQMN*, 8:28.

Chen, Y. and Nguyen, P. (2012). Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In Pointcheval, D. and Johansson, T., editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 502–519. Springer.

Cheon, J., Coron, J., Kim, J., Lee, M., Lepoint, T., Tibouchi, M., and Yun, A. (2013). Batch fully homomorphic encryption over the integers. In Johansson, T. and Nguyen, P., editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer Berlin Heidelberg.

Chung, K., Kalai, Y., and Vadhan, S. (2010). Improved delegation of computation using fully homomorphic encryption. In Rabin, T., editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 483–501. Springer Berlin Heidelberg.

Coron, J., Lepoint, T., and Tibouchi, M. (2014). Scale-invariant fully homomorphic encryption over the integers. In Krawczyk, H., editor, *Public-Key Cryptography - PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 311–328. Springer Berlin Heidelberg.

Coron, J., Naccache, D., and Tibouchi, M. (2012). Public key compression and modulus switching for fully homomorphic encryption over the integers. In Pointcheval, D. and Johansson, T., editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer Berlin Heidelberg.

Dahab, R., Galbraith, S., and Morais, E. (2015). Adaptive key recovery attacks on NTRU-based somewhat homomorphic encryption schemes. In Lehmann, A. and Wolf, S., editors, *Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 283–296. Springer International Publishing.

Dahab, R. and Morais, E. M. (2012). Encriptação homomórfica. In Santos, A. L. (Org.), S. A. O. M. C. O. and Gonçalves, P. A. S., editors, *Minicursos do XII Simpósio em Segurança da Informação e de Sistemas Computacionais. 12ed.*, pages 1–195.

Fiore, D., Gennaro, R., and Pastro, V. (2014). Efficiently verifiable computation on encrypted data. Cryptology ePrint Archive, Report 2014/202. `http://eprint.iacr.org/`.

Gennaro, R., Gentry, C., and Parno, B. (2010). Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Rabin, T., editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer Berlin Heidelberg.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA. ACM.

Howgrave-Graham, N. (2001). Approximate integer common divisors. In *CaLC*, pages 51–66.

Kim, J., Lee, M. S., Y., A., and Cheon, J. (2013). CRT-based fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/057. `http://eprint.iacr.org/`.

Lepoint, T. (2014). *Design and Implementation of Lattice-Based Cryptography*. PhD thesis, École Normale Supérieure and University of Luxembourg.

Loftus, J., May, A., Smart, N. P., and Vercauteren, F. (2011). On CCA-secure somewhat homomorphic encryption. In *In Selected Areas in Cryptography*, pages 55–72.

Morais, E. M., Aranha, D. F., and Dahab, R. AGCD-based CCA1-secure somewhat homomorphic encryption using verifiable computation. [Submitted].

Morais, E. M., Aranha, D. F., and Dahab, R. (2016a). Homomorphic encryption. Technical Report 02, Institution of Computing. `http://www.ic.unicamp.br/~reltech/2016/16-02.pdf`.

Morais, E. M., Aranha, D. F., and Dahab, R. (2016b). Lattice-based cryptography. Technical Report 01, Institution of Computing. `http://www.ic.unicamp.br/~reltech/2016/16-01.pdf`.

van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 24–43, Berlin, Heidelberg. Springer-Verlag.

