A Survey on Fully Homomorphic Encryption: An Engineering Perspective

PAULO MARTINS and LEONEL SOUSA, INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa, Portugal
ARTUR MARIANO, Institute for Scientific Computing, TU Darmstadt, Germany

It is unlikely that a hacker is able to compromise sensitive data that is stored in an encrypted form. However, when data is to be processed, it has to be decrypted, becoming vulnerable to attacks. Homomorphic encryption fixes this vulnerability by allowing one to compute directly on encrypted data. In this survey, both previous and current Somewhat Homomorphic Encryption (SHE) schemes are reviewed, and the more powerful and recent Fully Homomorphic Encryption (FHE) schemes are comprehensively studied. The concepts that support these schemes are presented, and their performance and security are analyzed from an engineering standpoint.

CCS Concepts: • **Security and privacy** → **Cryptography**; • **Theory of computation** → **Cryptographic primitives**; • **General and reference** → *Surveys and overviews*;

Additional Key Words and Phrases: Homomorphic encryption, ideals, learning with errors, number theory research unit, approximate greatest common divisor

ACM Reference format:
Paulo Martins, Leonel Sousa, and Artur Mariano. 2017. A Survey on Fully Homomorphic Encryption: An Engineering Perspective. *ACM Comput. Surv.* 50, 6, Article 83 (December 2017), 33 pages.
https://doi.org/10.1145/3124441

1 INTRODUCTION

Several billion devices are currently connected to the Internet, and this number will continue to grow. This is a consequence of not only more people becoming interested in consumer electronics but also more sensors and actuators being incorporated into everyday electronics, household appliances, and the general infrastructure. Since most of these devices are not able to process data locally, they will often upload it to a third party for processing. However, this data may be private, the third party may not be trustworthy, or both. Therefore, the data should be encrypted before it is transferred.

Cryptography refers generally to secret writing based on the use of ciphers. A cipher is a technique used to conceal information or reveal it, whereby a plaintext is converted into a ciphertext and vice versa. Most ciphers depend on an algorithm and a key. Whereas the algorithm establishes a way to transform data, the key is used as a parameter that modifies its behavior in a complex

This work was supported by Portuguese funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 and by the Ph.D. grant with reference SFRH/BD/103791/2014.
Authors' addresses: P. Martins and L. Sousa, Rua Alves Redol, 9, 1000-029 Lisboa, Portugal; emails: paulo.sergio@netcabo.pt, las@inesc-id.pt; A. Mariano, Scientific Computing Group, Technische Universität Darmstadt, Mornewegstr. 30, D-64293 Darmstadt, Germany; email: artur.mariano@sc.tu-darmstadt.de.

https://doi.org/10.1145/3124441

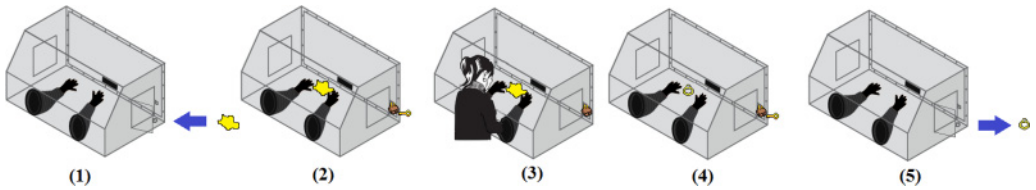


Fig. 1. A piece of gold is locked inside a glovebox, so that a worker may transform it into a ring. The ring is later removed when the glovebox is unlocked.

manner. Cryptosystems are usually classified in two categories: those where a common key is used for both ciphering and deciphering, based on private-key cryptography, and those supported in public-key cryptography, where a public key is used to cipher messages, and the ciphertext has to be deciphered by applying the corresponding secret key.

In order to perform operations on encrypted data, one must resort to Homomorphic Encryption (HE). This concept can be metaphorically explained by the jewelry shop problem (Gentry 2010), whose solution is represented in Figure 1. Alice, a shop owner, wanted her workers to assemble precious materials, such as gold and diamonds, into intricately designed rings and necklaces. However, she distrusted her workers, and thus did not want the workers to come in direct contact with the materials, since she was afraid they might steal them. In order to solve this problem, Alice used a transparent impenetrable glovebox. Alice would then open the box and store the raw materials inside. Afterward, she would lock the box using a key to which only she had access. As shown in Figure 1, the workers could use the gloves to assemble the rings and necklaces. Since the box was impenetrable, they could not have access to the precious materials. When the piece was finished, Alice could open the box and retrieve the result. In short, the workers were able to process the materials without having true access to them.

The jewelry box analogy represents several aspects of HE. The raw materials represent the data we do not want anyone else to have access to. Ciphering the data corresponds to locking it inside the impenetrable box, considering the lack of access as the lack of physical access and not visual access. The gloves portray the homomorphism of the encryption scheme, whereby it is possible to work on the precious material with gloves using equivalent techniques to those used barehanded. Finally, the rings and necklaces represent the desired function of the initial data.

While some schemes achieve HE partially (i.e., they only allow some specific operations on encrypted data (Rivest et al. 1978b; El Gamal 1985; Paillier 1999)), FHE has long been considered the holy grail of cryptography due to the multiple applications it has. In fact, researchers have been working on this problem since 1978 (Rivest et al. 1978a), but no such scheme was found for a long time. During the 30 years that followed, there was no evidence that FHE was even viable. There were nevertheless partial results, such as Boneh et al. (2005), which is homomorphic additive but allows at the same time one homomorphic multiplication. Gentry presented in his seminal work (Gentry 2009) in 2009 how to build the first FHE scheme. Multiple FHE schemes were presented since then, most of them focusing on improving efficiency.

This article introduces SHE and FHE schemes and their performance, while presenting the fundamental concepts of FHE, such as bootstrapping and modulus switching. All the research done so far in FHE is surveyed, mainly from an engineering perspective, refraining from delving too deep into the mathematics. Readers will learn the key principles of FHE, enough to understand the design of FHE systems, and remain with an extensive bibliography where they can find complementary material for all the aspects focused in this survey. Moreover, the motivation for HE will be provided with an analysis of its application to real-world problems. At the end of this survey, we will discuss the security of SHE/FHE schemes.

2 MOTIVATION: APPLICATIONS OF HOMOMORPHIC ENCRYPTION

The homomorphic evaluation of ciphertexts has a large number of applications. For example, it is useful for voting (Adida 2015), where users encrypt their vote, corresponding either to "0" or "1," for confidentiality, and all the ciphertexts are homomorphically summed. The sum is afterward decrypted and the result is obtained. The decryption key is split among a few partially trusted authorities who collaborate only for this single decryption. HE also underpins several other cryptographic protocols, such as two-party generation of Digital Signature Algorithm (DSA) signatures (MacKenzie and Reiter 2001), verifiable computing (Gennaro et al. 2010), and multiparty computation (Damgard et al. 2011).

Moreover, with HE, it is possible to implement a private database query system (Boneh et al. 2013). Using this system, when a client issues a query to the database, the server is not able to understand what has been queried. Additionally, the user learns nothing but the result of the query. Using HE, it is also possible to delegate the statistical analysis of sensitive data collected by sensors to an untrusted third party (Wu and Haven 2012; Naehrig et al. 2011).

In another application (Naehrig et al. 2011), a company may want to use detailed information about a potential customer so that the advertisements that he or she sees are of his or her interest. This information might be obtained from the consumer's cell phone, by uploading common words in his or her e-mails, recently visited websites, and so forth to the server of the advertising company. Since this information is private, it is convenient to cipher it with the consumer's public key. If the advertisements are also ciphered with the same key, it is possible to homomorphically compute which advertisement should be displayed. The resulting ciphertext is then transferred back to the user, who decrypts it using the secret key.

In some situations (Naehrig et al. 2011), it may be necessary to protect both the plaintext and the algorithms that process it. An example is that of Magnetic Resonance Imaging (MRI) machines, where the most expensive part is related to the algorithm used to analyze the magnetic resonance data. Hardware devices are used to protect this algorithm and destroy the program before being examined by a third party. With HE, it is possible for an MRI producer to centralize data computation, reducing the cost of the machines while preserving the patient private data.

3 NOTATION

Throughout this article, the notation $x = y \pmod{q}$ is used to express that $x - y$ is divisible by the modulus q. When the modulus is clear from the context, the notation is omitted. For an integer or rational number x and an integer q, we denote the operation of mapping x to the interval $(-q/2, q/2]$ by adding or subtracting a multiple of q as $x \bmod q$ or $[x]_q$ and refer to this expression as reducing x modulo q. The rounding of the rational number x to the nearest integer is denoted by $\lfloor x \rceil$, and the fractional part of x by $[x] = x - \lfloor x \rceil$. We denote the XOR of two bits by $\oplus$, the AND by $\wedge$, and the NOT of a bit b by $\overline{b}$. The same notations are applied to vectors to express that the same operation is applied to all entries of the vector.

We denote vectors by a right-pointing arrow over its name (e.g., $\vec{x}$) and matrices by capital letters. We assume a row-major representation and denote vector-matrix multiplication of $\vec{x} \in R^n$ and $M \in R^{n \times m}$, for some ring R, as $\vec{x} \times M$. The dot product of $\vec{x}, \vec{y} \in R^n$ is denoted by $\langle \vec{x}, \vec{y} \rangle$, the tensor product of $\vec{x}, \vec{y} \in R^n$ by $\vec{x} \otimes \vec{y}$, and the Euclidean norm of $\vec{x} \in \mathbb{Q}^n$ by $||\vec{x}||$. A ring R refers to a set that is closed under two binary operations $\times$ and $+$.

We represent in lowercase elements x of rings R, that is, $x \in R$. Two examples of rings are (1) the integer polynomials $\mathbb{Z}[t]$, which can be added and multiplied with each other, and (2) the quotient ring $\mathbb{Z}[t]/(f(t))$, where two elements are congruent if their difference is a multiple of $f(t)$. An ideal I of a ring R is a subset of R, such that $\forall_{x \in R} \forall_{y \in I} x \times y \in I$. The set of even integers $(2\mathbb{Z})$,

for instance, is an ideal in the ring $\mathbb{Z}$. Sometimes principal ideals will be used. Given $y \in R$, the principal ideal Y generated by y is $Y = (y) = \{x \times y | x \in R\}$. Moreover, if $\mathbb{F}$ is a field that contains R, the inverse of $I \subseteq R$ with respect to $\mathbb{F}$ is $I^{-1} = \{z \in \mathbb{F} | \forall y \in I, z \times y \in R\}$. In the particular case of a principal ideal (y), its inverse in $\mathbb{F}$ is also a principal ideal generated by its inverse y^{-1} over the field $\mathbb{F}$, that is, (y^{-1}). For an integer q, R_q is used to denote R/qR. For $x \in \mathbb{Z}[t]/(f(t))$, we denote by $|x|$ the Euclidean norm of the vector whose entries are the coefficients of x. In addition, $x \leftarrow \chi$ is used to express that x is picked from distribution χ, or $x \leftarrow S$ is used to denote that x is randomly drawn from the set S using a uniform distribution.

Furthermore, we use $y = \text{CRT}_{(q_1,\ldots,q_n)}(x_1, \ldots, x_n)$ for pairwise coprime $q_1, \ldots q_n \in \mathbb{Z}$ and $x_i \in \mathbb{Z}_{q_i}$ to denote the integer $y \in (-q/2, q/2]$, where $q = \prod_{i=1}^{n} q_i$, congruent to x_1 modulo q_1, x_2 modulo q_2, and so forth. Additionally, we use LCM to denote the least common multiple. Moreover, we define the m^{th} cyclotomic polynomial $\Phi_m(t)$ for $m \in \mathbb{Z}$ and $m \geq 0$ to be $\Phi_m(t) = \prod_{\zeta}(t - \zeta)$ with ζ ranging over the primitive m^{th} roots of unity.

Finally, we define an arithmetic circuit C over the field $\mathbb{F}$ and the set of variables T (usually $T = \{t_1, \ldots, t_n\}$) to be a directed acyclic graph as follows. The vertices of C are called gates. Every gate in C of in-degree 0 is labeled by either a variable from T or a field element from $\mathbb{F}$. Every other gate in C is labeled by either $\times$ or $+$ and has in-degree 2. We further denote by f_g the polynomial in $\mathbb{F}[T_g]$ computed by the gate g in C, where T_g is the set of variables that occur in the circuit C_g, corresponding to the subcircuit of C rooted at g. We take the degree of f_g to be the degree of gate g. The degree of C is the maximal degree of a gate in C.

4 CLASSICAL HOMOMORPHIC ENCRYPTION SCHEMES

We begin by giving a precise definition of public-key cryptography, following Gentry (2010). A public-key encryption scheme $\mathcal{E}$ is composed of a set of polynomial-time methods $(\textbf{KeyGen}_{\mathcal{E}}, \textbf{Encrypt}_{\mathcal{E}}, \textbf{Decrypt}_{\mathcal{E}})$, of which $\textbf{KeyGen}_{\mathcal{E}}$ and $\textbf{Encrypt}_{\mathcal{E}}$ are probabilistic, which satisfies the following:

- $\textbf{KeyGen}_{\mathcal{E}}$ outputs a public key (**pk**), which is used for encryption, and the corresponding secret key (**sk**), used for decryption; taking the security parameter λ as an argument.
- $\textbf{Encrypt}_{\mathcal{E}}$ uses the public key **pk** to map a plaintext into a ciphertext.
- Conversely, $\textbf{Decrypt}_{\mathcal{E}}$ maps the ciphertext back into plaintext, using the corresponding secret key **sk**. However, should decryption fail, a special symbol $\perp$ is returned.

A homomorphic encryption scheme integrates a fourth algorithm, herein denoted by $\textbf{Evaluate}_{\mathcal{E}}$. To this scheme we can associate a set $\mathcal{F}_{\mathcal{E}}$ containing all functions that can be homomorphically evaluated by the scheme; that is, for $f \in \mathcal{F}_{\mathcal{E}}$, $\textbf{Evaluate}_{\mathcal{E}}(\textbf{pk}, f, c_0, \ldots, c_{t-1})$, where $c_0, \ldots, c_{t-1}$ correspond to encryptions of $m_0, \ldots, m_{t-1}$ under **sk**, outputs a ciphertext c, such that $\textbf{Decrypt}_{\mathcal{E}}(\textbf{sk}, c) = f(m_0, \ldots, m_{t-1})$. However, a scheme where $\textbf{Evaluate}_{\mathcal{E}}$ returns f and $c_0, \ldots c_{t-1}$ so that the function is evaluated during decryption satisfies the previous definition. To prevent this, we need to force the output to be at most s bits long, where s is a polynomial $s = \text{pol}(\lambda)$ (Brakerski and Vaikuntanathan 2011a). This property is called compactness (Brakerski and Vaikuntanathan 2011a).

While this survey is mainly focused on the most recent developments on homomorphic cryptography, we start by providing an overview of three classical homomorphic schemes: Paillier, which allows homomorphic additions, and RSA and ElGamal, which allow homomorphic multiplications. In particular, we put forward an analysis of the homomorphic properties of these schemes, which is a valuable way of introducing the topic.

Cryptosystem 1: Paillier

During the $\mathbf{KeyGen}_{\mathcal{E}}$ procedure, two different prime numbers with identical bit-length r, s are randomly generated. We take $q = rs$ to be the public-key, and the private-key corresponds to the tuple (λ, μ), where $\lambda = \mathbf{LCM}(r-1, s-1)$ and $\mu = \lambda^{-1} (\mathrm{mod}\ q)$.
A ciphertext c of a plaintext $m \in \mathbb{Z}_q$ corresponds to $c = (1+q)^m u^q (\mathrm{mod}\ q^2)$, where $u \leftarrow \mathbb{Z}_q^*$. We define $l : q\mathbb{Z} + 1 \mapsto \mathbb{Z}$, $y \mapsto l(y) = \frac{y-1}{q}$. Decryption is performed by first computing $l(c^\lambda \bmod q^2)$, and multiplying the result by μ modulo q. We note that by the binomial theorem $(1+q)^x = \sum_{k=0}^{x} \binom{x}{k} q^k = 1 + qx (\mathrm{mod}\ q^2)$, therefore if $y = (1+q)^x (\mathrm{mod}\ q^2)$, the discrete logarithm of y with respect to $1+q$ can be efficiently computed using $l(y)$. Decryption works correctly, since by Carmichael's function [Erdös et al. 1991] $u^{q\lambda} = 1 (\mathrm{mod}\ q^2)$, and further $(1+q)^{m\lambda} = 1 (\mathrm{mod}\ q)$, thus $\mu l(c^\lambda \bmod q^2) = \mu m \lambda = m (\mathrm{mod}\ q)$.
Denoting $g = q + 1 (\mathrm{mod}\ q^2)$, the multiplication of two ciphertexts $c_1 = g^{m_1} u_1^q (\mathrm{mod}\ q^2)$ and $c_2 = g^{m_2} u_2^q (\mathrm{mod}\ q^2)$ results in $c_3 = g^{m_1+m_2} (u_1 u_2)^q (\mathrm{mod}\ q^2)$ that is a valid encryption of $m_1 + m_2$.

Paillier is a public-key cryptosystem (Paillier 1999) that relies on the Decisional Composite Residuosity Assumption (DCRA). DCRA states that given a composite integer q and an integer $x \in \mathbb{Z}_{q^2}$, it is hard to decide whether there is a $y \in \mathbb{Z}_{q^2}$ such that $x \equiv y^q (\mathrm{mod}\ q^2)$. The procedures for this cryptosystem are presented in Cryptosystem 1. As made evident in Cryptosystem 1, Paillier is homomorphically additive: given encryptions of m_1 and m_2, we are able to compute an encryption of $m_1 + m_2$ without having access to the secret key. However, it is not known how to produce an encryption of $m_1 \times m_2$ from c_1 and c_2, and thus the scheme is not homomorphically multiplicative.

Cryptosystem 2: ElGamal

Let g be a generator of the cyclic group $\mathbb{G}$ with order q. Further assume that the Decisional Diffie-Hellman (DDH) assumption holds on $\mathbb{G}$. We select a random value $s \leftarrow \mathbb{Z}_q$ to be the secret-key with corresponding public-key $h = g^s$.
To cipher a message $m \in \mathbb{G}$, we produce the vector (c_1, c_2), where $c_1 = g^u$, with u selected at random from $\mathbb{Z}_q$, and $c_2 = mh^u$. Decryption of the ciphertext comprises computing $m = c_2 c_1^{-s}$.
Component-wise multiplication of two ciphertexts $(c_{1,1} = g^{u_1}, c_{1,2} = m_1 h^{u_1})$ and $(c_{2,1} = g^{u_2}, c_{2,2} = m_2 h^{u_2})$ results in a tuple $(c_{3,1}, c_{3,2})$, such that $c_{3,1} = g^{u_1+u_2}$ and $c_{3,2} = m_1 m_2 h^{u_1+u_2}$ that constitutes a valid encryption of $m_1 m_2$.

El Gamal (1985) is a public-key cryptosystem that is supported on the Decisional Diffie-Hellman (DDH) problem: if $\mathbb{G}$ is a group with generator g, no one can distinguish distributions $\langle g^x, g^y, g^{xy} \rangle$ and $\langle g^x, g^y, g^z \rangle$ with more than negligible probability, where x, y, and z are integers chosen randomly between 1 and the order of $\mathbb{G}$. The procedures proposed by El Gamal can be found in Cryptosystem 2. It can be noted that El Gamal is multiplicatively homomorphic: given encryptions of m_1 and m_2, we can get a ciphertext encrypting $m_1 m_2$ without accessing the secret key. Unfortunately, it is not possible to compute the encryption of $m_1 + m_2$ given the ciphertexts c_1 and c_2, and thus the scheme is multiplicatively homomorphic but not additively homomorphic.

The Rivest-Shamir-Adleman (RSA) scheme (Rivest et al. 1978b), in its textbook format, is depicted in Cryptosystem 3 and is related to the problem of factoring large integers. We can see that this cryptosystem is multiplicatively homomorphic, but not additively homomorphic. However,

Cryptosystem 3: Rivest-Shamir-Adleman

Two primes r and s of similar bit-length are randomly generated during $\mathbf{KeyGen}_{\mathcal{E}}$. The public-key corresponds to a tuple (q, e), with $q = rs$, $\gcd(e, \phi(q)) = 1$, and $\phi(q) = (r-1)(s-1)$. The corresponding secret-key is $d = e^{-1} (\bmod\ \phi(q))$, i.e. the modular inverse of e modulo $\phi(q)$.
Encryption of a message m leads to $c = m^e (\bmod\ q)$, while decryption corresponds to the exponentiation of the ciphertext to the d^{th} power modulo q. Decryption works correctly because the exponents operate modulo $\phi(q)$ and cancel each other. Multiplication of two ciphertexts c_1 and c_2 encrypting m_1 and m_2, respectively, results in $c_3 = (m_1 m_2)^e \bmod\ m$, which corresponds to the encryption of $m_1 m_2$.

this homomorphically multiplicative scheme is not used in practice, since textbook RSA is deterministic. Hence, it is insecure, because an attacker who knows the correspondence between a plaintext and a ciphertext will always be able to recognize when the same plaintext has been encrypted.

Before the uncovering of FHE schemes, in 2009, some already made proposals supported a bounded amount of additions and multiplications simultaneously, contrarily to the aforementioned schemes. From these early publications, we highlight the work of Boneh et al. (2005). It is based on the El Gamal cryptosystem but instantiated over Elliptic Curves (ECs). An EC is a mathematical variety with an algebraic group. This group has a binary operation named point addition. This operation is applied repeatedly to perform point multiplication, and this latter operation replaces exponentation in El Gamal. Using Boneh et al. (2005), the addition of the ciphertexts results in the addition of the underlying plaintexts. Also, a single homomorphic multiplication is possible using bilinear pairings that convert points from ECs to Finite Fields (FFs). Since no bilinear pairings are known that can be applied to FFs, no further multiplications are possible.

5 BRIEF INTRODUCTION TO LATTICE- AND RING-BASED CRYPTOGRAPHY

In this section, we briefly review the theoretical constructs necessary to support lattice- and ring-based cryptography. It adopts the formal definition of rings, ideals, and principals given in Section 3.

A lattice is a discrete additive subgroup of $\mathbb{R}^m$. Equivalently, it corresponds to the vector space generated by all linear combinations with integer coefficients of a set $\mathcal{B} = \{\vec{b}_0, \ldots, \vec{b}_{n-1}\}$, with $\vec{b}_i \in \mathbb{R}^m$, of linearly independent vectors:

$$\mathcal{L}(\mathcal{B}) = \left\{ \sum_{i=0}^{n-1} z_i \vec{b}_i : z_i \in \mathbb{Z} \right\}. \tag{1}$$

The set $\mathcal{B}$ is also associated with a matrix B, having the vectors $\vec{b}_i$ as rows. The previous expression can thus be written as $\mathcal{L}(B) = \{\vec{z} \times B : \vec{z} \in \mathbb{Z}^n\}$.

Lattices have an infinite number of bases for $n \geq 2$. Furthermore, any two matrices B_1 and B_2 associated with the same lattice are related by an integer matrix U with $|\det(U)| = 1$, that is, $B_1 = U \times B_2$. Therefore, $\det(B_1) = \pm \det(B_2)$ and the absolute value of the determinant is the same for all the bases of $\mathcal{L}$. This value is often denoted by $\det(\mathcal{L})$. Every base B has a corresponding half-open parallelepiped $\mathcal{P}(B) \leftarrow \{\sum_{i=0}^{n-1} z_i \vec{b}_i : z_i \in (-1/2, 1/2]\}$. A graphical representation of part of a lattice and its fundamental parallelepiped can be found in Figure 2.

Every lattice induces a congruence relation, wherein two vectors are congruent ($\vec{x} = \vec{y} (\bmod\ \mathcal{L})$) if their difference is in the lattice ($\vec{x} - \vec{y} \in \mathcal{L}$). The reduction of a vector $\vec{y}$ modulo a lattice base

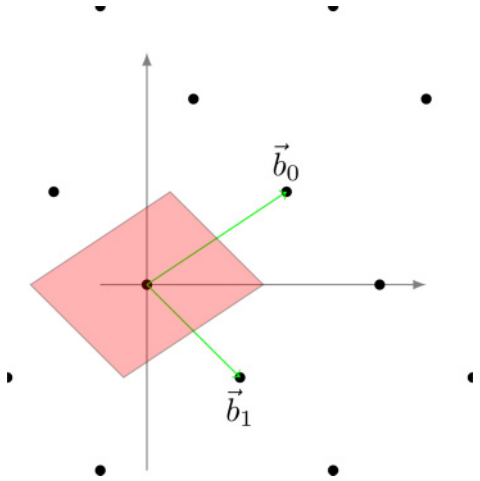


Fig. 2. Representation of a section of a lattice in $\mathbb{R}_2$ (black points), with its base vectors in green and its fundamental parallelepiped in red. The black dots continue indefinitely in the lattice.

B, $\vec{x} = \vec{y} \bmod B$, corresponds to determining $\vec{x} \in \mathcal{P}(B)$ congruent with $\vec{y}$. This operation can be computed as $\vec{x} = \vec{y} - \lfloor \vec{y} \times B^{-1} \rceil \times B = [\vec{y} \times B^{-1}] \times B$.

The Hermite Normal Form (HNF) base, which is unique to every lattice, corresponds to a base H, such that $\forall i < j\, h_{i,j} = 0$; $\forall j\, h_{j,j} > 0$; and $\forall i > j\, h_{i,j} \in (-h_{j,j}/2, +h_{j,j}/2]$. The HNF can be efficiently computed from any basis B of a lattice (Gentry 2009). It is therefore the "least revealing" base. This property makes it often a good choice for the public key of Lattice-Based Cryptosystems (LBCs).

We will mostly be working with rings of the type $R = \mathbb{Z}[t]/(f(t))$, for some polynomial $f(t)$. If we consider the elements of an ideal I of R as vectors, I forms a lattice since it is closed under addition.

LBCs are usually supported on either the Closest Vector Problem (CVP), the Shortest Vector Problem (SVP), or the General Learning with Errors (GLWE), the definitions of which follow (Bernstein et al. 2008; Brakerski et al. 2012), and can be instantiated for any norm. The Learning with Errors (LWE) and Ring Learning with Errors (RLWE) problems are captured in Definition 3 and correspond to GLWE when $m = 1$ and $n = 1$, respectively.

Definition 1 (CVP). Given a base $B \in \mathbb{R}^{n \times m}$, and $\vec{y} \in \mathbb{R}^m$, find $\vec{x} \in \mathcal{L}(B)$, such that $||\vec{y} - \vec{x}|| = \min_{\vec{z} \in \mathcal{L}(B)} ||\vec{y} - \vec{z}||$.

Definition 2 (SVP). Given a base $B \in \mathbb{R}^{n \times m}$, find $\vec{x} \in \mathcal{L}(B)$, such that $||\vec{x}|| = \min_{\vec{z} \in \mathcal{L}(B) \setminus \vec{0}} ||\vec{z}||$.

Definition 3 (GLWE). Let $n, m, q \in \mathbb{Z}$; let $R = \mathbb{Z}[t]/(\Phi_m(t))$, $R_q = R/(qR)$; and let χ be a distribution over R (typically a Gaussian distribution). Given arbitrarily many samples $(\vec{x}_i, y_i) \in R_q^{n+1}$, where $y_i = \langle \vec{x}_i, \vec{s} \rangle + e_i$, with $\vec{x}_i, \vec{s} \leftarrow R_q^n$ sampled uniformly and $e_i \leftarrow \chi$, find $\vec{s}$.

We may intuitively establish a connection between the LWE problem and lattices (a similar rationale can be conducted for the RLWE). First we define the lattice $\mathcal{L}(B)$, where the matrix $B \in \mathbb{Z}^{n \times t}$ has t $\vec{x}_i$ samples as columns. If we were able to compute the closest vector, $\vec{y'}$ to $\vec{y} = (y_0, \ldots, y_{t-1})$ in the lattice, then solving the system $\vec{s} \times B = \vec{y'}$ would provide us with a solution to the LWE problem.

6 FULLY HOMOMORPHIC PUBLIC-KEY CRYPTOGRAPHY

Cryptosystems can be characterized according to the extent up to which they are homomorphic, leading to the definitions of the SHE and FHE schemes that follow, which were inspired by Gentry (2010). With these schemes, one specifies the computation to be performed not using sequential programs, but rather using arithmetic circuits or networks, where signals pass through a cascade of logical gates (cf. Section 3).

Definition 4 (SHE). An encryption scheme $\mathcal{E}$ is somewhat homomorphic when it is compact and $\mathcal{F}_{\mathcal{E}}$ contains a nonnull subset of all arithmetic circuits over $\mathbb{F}_2$.

Definition 5 (FHE). An encryption scheme $\mathcal{E}$ is fully homomorphic when it is compact and $\mathcal{F}_{\mathcal{E}}$ contains all arithmetic circuits over $\mathbb{F}_2$.

Modern cryptosystems are typically semantically secure: an attacker is not able to determine if a ciphertext encrypts a certain plaintext even if he or she has had access to one encryption of that plaintext before. This is achieved using a probabilistic encryption scheme, where when one encrypts a plaintext, one may get many values, but when decrypting all these values, they collapse to the same plaintext.

In his breakthrough work, Gentry (2009) started by proposing a semantically secure SHE based on ideals. Plaintexts are encrypted as a perturbation to an element of the ideal, such that it is only possible to compute the nearest ideal element to a ciphertext given the secret key of the scheme. To produce a semantically secure scheme, the perturbations should not be deterministic because otherwise the ciphertexts will be deterministic too. Therefore, it was proposed to add a random "noise" to the value being ciphered. The decryption function filters out the noise and treats each point as if it were located in the nearest unperturbed location. However, when homomorphic operations are applied to the ciphertexts, their noise grows. The perturbed point thus wanders farther from its "correct" position until the decryption function starts associating it with another point.

Roughly speaking, each homomorphic addition adds the underlying noises, and each multiplication multiplies them. Hence, noise growth limits the amount of operations that can be accomplished. This limit could be evaded if, when the noise begins to approach the critical threshold, the data could be decrypted and reciphered, thereby resetting the noise to its original low level. However, this would require access to the secret key, defeating the purpose of FHE.

A solution to this problem comes with bootstrapping, a technique wherein a cryptosystem evaluates its own decryption circuit homomorphically. If this is possible, one can integrate in the public key an encrypted version of the secret key, thereby making it possible to reset the noise level homomorphically. The concept of bootstrapping is illustrated in Figure 3. Concretely, suppose that one generates two secret/public-key pairs and encrypts the bits of the first secret key using the second public key, making these ciphertexts public. An evaluator would proceed as follows to refresh the noise level of a ciphertext associated with the first public key. He or she would start by ciphering the bits of this ciphertext with the second public key. Afterward, the decryption circuit would be evaluated. This circuit has inputs related to ciphertext bits and the secret-key bits. Since we have encrypted versions of both, the decryption circuit can be homomorphically evaluated. However, it should be noted that even though the decryption operation eliminates the noise of the ciphertext with respect to the first public key, noise is produced when evaluating the circuit with respect to the second public key. In order to make some progress, the final noise should be less than the initial, and should enable performing at least one more operation.

One can now apply this technique iteratively to get an FHE scheme. If one wants to evaluate circuits with n levels of gates, one would start by generating $n + 1$ public keys of the underlying SHE scheme and encrypt the bits of each secret key at level i, $i \in \{0, \ldots, n-1\}$, with the public

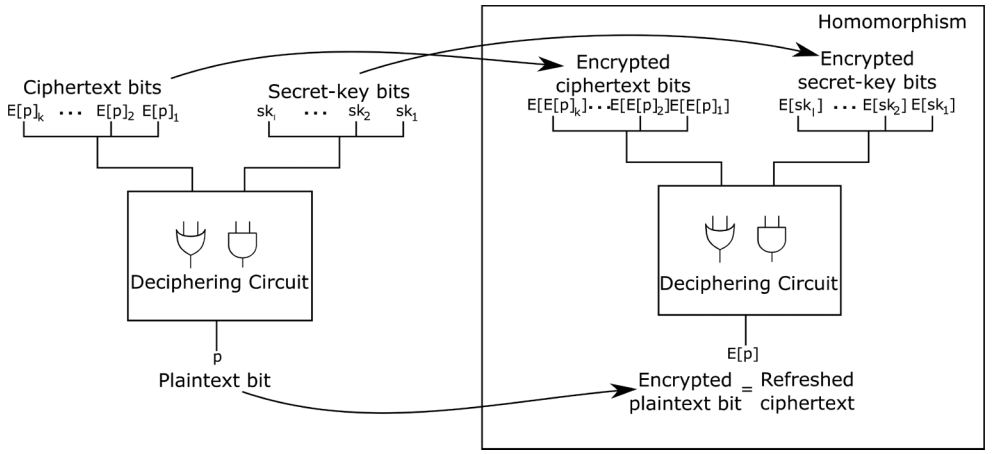


Fig. 3. The decryption operation applied to the ciphertext bits and the secret-key bits on the left, and the homomorphic decryption of their encrypted counterparts on the right.

key at level $i + 1$. One would then, for each of the two inputs of a gate at level i, encrypt the bits of the input ciphertext of level $i - 1$ using the public key of level i. Evaluating the decryption circuit with the encrypted secret key of level $i - 1$ with respect to the public key of level i for both inputs produces refreshed ciphertexts, to which a homomorphic addition or multiplication can be applied. Finally, the result of the operation is output to the next level.

An inconvenience with the previous description is that $\mathbf{KeyGen}_{\mathcal{E}}$ depends linearly on the depth of the circuits that can be homomorphically evaluated. If one assumes the scheme to be circular secure, an alternative is to provide a single key pair, while encrypting the secret key with the corresponding public key. Now, all circuit levels are associated with the same key pair, and therefore arbitrary circuits may be evaluated without specifying their maximum depth beforehand.

Gentry's first FHE scheme motivated a large amount of research in the area, mainly with the purpose of improving the efficiency of the original scheme. A timeline of this research is presented in Figure 4, with the most relevant references. Even though Gentry's scheme was based on ideals, it has been ported to schemes using different hardness assumptions, which are depicted in timelines with different colors, namely, Approximate Greatest Common Divisor (AGCD), LWE, and Number Theory Research Unit (NTRU). These schemes have enabled the use of techniques such as modulus switching and scale invariance that allow a slow growth of the noise associated with the ciphertexts, removing the need for bootstrapping. The use of batching, wherein each ciphertext encrypts several messages that can be processed in parallel, is also relevant to improve performance. These techniques will be described in detail in the following sections.

6.1 Ideal-Based FHE

The work of Gentry (2009) describes an SHE encryption scheme similar to the Goldreich-Goldwasser-Halevi (GGH) cryptosystem (Goldreich et al. 1997) but supported on ideals. The secret key of a GGH scheme corresponds to a "good" basis of a lattice, B, whereas the public key is a "bad" basis of the same lattice, H. The main difference between the two bases is that the former has nearly orthogonal vectors, while in the second vectors are quite skewed. The ciphertext of a message $\vec{m}$, encoded as a small error vector, is $\vec{c} \leftarrow \vec{m} \bmod H$. It can be seen that when $\vec{m}$ has a small enough norm, it can be computed from the difference between $\vec{c}$ and the closest lattice vector. With B, the key holder is able to compute the closest lattice point to $\vec{c}$, and from there calculate the value of

Fig. 4. Evolution of FHE.

$\vec{m}$. On the other hand, the basis parallelepiped $\mathcal{P}(H)$ will be very skewed, and therefore unfit to compute the closest lattice point.

Smart and Vercauteren, in 2009, tried for the first time to implement an FHE similar to Gentry's but supported on principal ideals (Smart and Vercauteren 2009). Although the underlying SHE scheme was implemented, it did not achieve bootstrappability . In 2011, Gentry and Halevi showed a number of optimizations that enabled a fully functioning implementation of FHE (Gentry and Halevi 2011). Their scheme corresponds to an application of GGH to ideal lattices of the ring $R = \mathbb{Z}[t]/(f(t))$, where $f(t) = t^d + 1$ with d a power of two. The secret and public keys, B and H, correspond to "good" and "bad" bases of an ideal lattice I, respectively. Herein, we view elements of R both as elements of polynomial rings and vector. The underlying lattice corresponds to the principal ideal $Y = (\vec{y})$ generated by a random vector $\vec{y}$ in some d-dimensional cube. We recall that $Y = \{\vec{y} \times \vec{x} | \vec{x} \in R\}$, where we see $\vec{y}$ both as a vector and as a polynomial $y(t) = \sum_{i=0}^{d-1} y_i t^i$. In particular, Y has a "rotational" base defined as $\{\vec{y}_i = y(t) \times t^i (\text{mod } f(t)) | i \in \{0, \ldots, d-1\}\}$. We

notice that $y(t) \times t = -y_{d-1} + y_0 t + \cdots + y_{d-2} t^{d-1} (\text{mod } f(t))$. If we repeat this operation for all powers of t, we get the rotational base of the lattice:

$$B = \begin{pmatrix} y_0 & y_1 & y_2 & \cdots & y_{d-1} \\ -y_{d-1} & y_0 & y_1 & \cdots & y_{d-2} \\ -y_{d-2} & -y_{d-1} & y_0 & \cdots & y_{d-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -y_1 & -y_2 & -y_3 & \cdots & y_0 \end{pmatrix}. \tag{2}$$

B is used as the secret key. Notwithstanding, Gentry and Halevi used an optimized representation of B that we will describe later in this section. Furthermore, Y is restricted such that its HNF has the following format:

$$H = \text{HNF}(B) = \begin{pmatrix} q & 0 & 0 & \cdots & 0 \\ -r & 1 & 0 & \cdots & 0 \\ -[r^2]_q & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -[r^{n-1}]_q & 0 & 0 & \cdots & 1 \end{pmatrix}. \tag{3}$$

If H does not have the desired format, a new vector $\vec{y}$ is generated. This restriction is met with a probability of roughly 0.5. The HNF is very skewed, since all vectors are close to being in the same direction, resulting in a very small decryption radius. Thus, it is used as the public key. Moreover, one can reconstruct the matrix with just q and r, where $q = \det(H)$.

Similarly to GGH, a message is embedded in a perturbation that is added to an element of Y. More specifically, a bit $m \in \{0, 1\}$ is encrypted by first computing $\vec{x} = 2\vec{u} + m\vec{e}_1$, where $\vec{u}$ is a random vector with entries $0, \pm 1$. The ciphertext corresponds to the reduction of $\vec{x}$ modulo H, that is, $\vec{c} = \vec{x} \bmod H$. Gentry and Halevi proved that $\vec{c} = (c, 0, \ldots, 0)$, where $c = [x(r)]_q = [m + 2\sum_{i=0}^{d-1} u_i r^i]_q$.

In order to decrypt $\vec{c}$, the ciphertext is reduced modulo the secret key, that is, $\vec{x} = \vec{c} \bmod B = [\vec{c} \times B^{-1}] \times B$. The inverse of B is a matrix of the form $B^{-1} = q^{-1}B'$, such that B' has the same format as B; that is, the first row is $(b'_0, b'_1, \ldots, b'_{d-1})$, the second $(-b'_{d-1}, b'_0, \ldots, b'_{d-2})$, and so on. As an optimization, Gentry and Halevi also showed that $[cb'_i]_q = mb'_i \bmod 2$, for every i. Thus, the decryption step can be simplified to $m = [cb'_i]_q \bmod 2$, for an odd b'_i, and the value b'_i is now used as the secret key.

This scheme shows some homomorphic capabilities. Every ciphertext is of the form $\vec{c} = \vec{y} + \vec{x}$, where $\vec{y}$ is in the lattice Y. If we add $\vec{c}_1 = \vec{y}_1 + \vec{x}_1$ and $\vec{c}_2 = \vec{y}_2 + \vec{x}_2$, we get

$$\vec{c}_3 = \underbrace{\vec{y}_1 + \vec{y}_2}_{\vec{y}_3} + \underbrace{\vec{x}_1 + \vec{x}_2}_{\vec{x}_3}, \tag{4}$$

and $\vec{y}_3$ is also in Y. Since both $\vec{x}_1$ and $\vec{x}_2$ encode a bit along their first coefficient, the bit encoded by $\vec{x}_3$ becomes the sum of these two bits modulo 2, that is, the XOR of the two bits. Similarly, their product is

$$\vec{c}_4 = \underbrace{\vec{y}_1 \times (\vec{y}_2 + \vec{x}_2) + \vec{x}_1 \times \vec{y}_2}_{\vec{y}_4} + \underbrace{\vec{x}_1 \times \vec{x}_2}_{\vec{x}_4}. \tag{5}$$

From the definition of an ideal, $\vec{y}_4 \in Y$. Again, since both $\vec{x}_1$ and $\vec{x}_2$ encode a bit along their first coefficient, the bit encoded by $\vec{x}_4$ will become the product of these two bits modulo 2, which corresponds to the logical AND of the two bits. However, we can see that with every addition and multiplication the size of $\vec{x}$ grows, and as more operations are performed, it eventually becomes larger than the decryption radius of B. This problem was solved with bootstrapping.

Bootstrapping corresponds to the technique of reducing the noise associated with a ciphertext by homomorphically decrypting it. The previously described SHE scheme is not bootstrappable, since the degree of the decryption function is too high to be homomorphically computed. Gentry (2009) showed how to "squash" the decryption procedure to enable its homomorphic evaluation. We recall that Gentry and Halevi's scheme secret key corresponds to an integer b'_i. The solution is to add a large set of integers $\mathcal{S} = \{x_i | i \in \{0, \ldots, S-1\}\}$ to the public key such that a sparse subset of $\mathcal{S}$ adds up to b'_i modulo q, as well as an encryption of the new equivalent of the secret key: a vector of encryptions of $\vec{\delta} = (\delta_0, \ldots, \delta_{S-1})$ corresponding to the sparse subset of $\mathcal{S}$. This means that $b'_i = \sum_{i=0}^{S-1} \delta_i \times x_i (\text{mod } q)$.

Then, given a ciphertext c, one can compute all the integers $y_i = [cx_i]_q$. The decryption function is adapted to a low-degree circuit as

$$\textbf{Decrypt}_{\mathcal{E}}(\textbf{sk}, c) = \left[\sum_{i=0}^{S-1} \delta_i y_i \right]_q \mod 2. \tag{6}$$

This is quite useful, since the largest part of the procedure can be computed in the clear, and $\vec{\delta}$ is afterward used to finish the procedure with low homomorphic complexity. With this technique, and assuming the hardness of the Sparse Subset Problem (SSSP), it is now possible to implement an FHE scheme.

6.1.1 Performance. Figure 5 shows the performance reported in the literature for several implementations of Gentry's FHE scheme. Due to security reasons, the scheme requires handling integers with hundreds of thousands to millions of bits. This degrades efficiency in a twofold manner. First, public-key sizes are prohibitively large, ranging from 69MB for $d = 2,048$ to 2.25GB for $d = 32,768$, where d corresponds to the lattice's dimension. Second, one has to multiply very large integers, which becomes the main bottleneck in these systems.

The implementation in Gentry and Halevi (2011) already handles several of these issues, in particular by using an optimized version of Equation (6) that decreases the **pk** size, since the public key has data that supports its homomorphic evaluation. Nevertheless, its large size, as well as the large $\textbf{KeyGen}_{\mathcal{E}}$ execution time, is mostly related to the ciphering of the bits of the secret key. Moreover, a large part of the execution time of the $\textbf{Recrypt}_{\mathcal{E}}$ operation, corresponding to the homomorphic evaluation of the decryption algorithm, is related to the large integer multiplications required. With this concern in mind, Wang et al. have proposed to accelerate integer multiplications (Wang et al. 2012) using Graphics Processing Units (GPUs). GPUs excel at processing highly parallel code with few divergences. Thus, Strassen's Fast Fourier Transform (FFT)-based multiplication algorithm (Schönhage and Strassen 1971) was applied to perform large integer multiplications, and subsequently Barrett's modular reduction (Barrett 1987) was used to reduce the result. Barrett's algorithm precomputes a scaled inverse of the modulo q, so that modular reductions can be efficiently computed using only multiplications and shifts. Later, Wang and Huang implemented the same scheme on a Field-Programmable Gate-Array (FPGA) (Wang and Huang 2013), namely, a Stratix-V, and reported a twofold acceleration relative to the GPU, at lower power consumption. A similar approach was applied in Doröz et al. (2015) to derive an Application-Specific Integrated Circuit (ASIC) accelerator for Gentry's scheme. Although an older manufacturing technology of 90nm is used in comparison to the previously referred devices, Doröz et al. achieve a better performance for encryption and recryption.

Following a different optimization strategy, Dalibard has shown how to accelerate the encryption and recryption functions, by parallelizing them. When encryption is performed for the first time, the computation of the $[r^i]_q$ is distributed among several processing nodes; afterward,

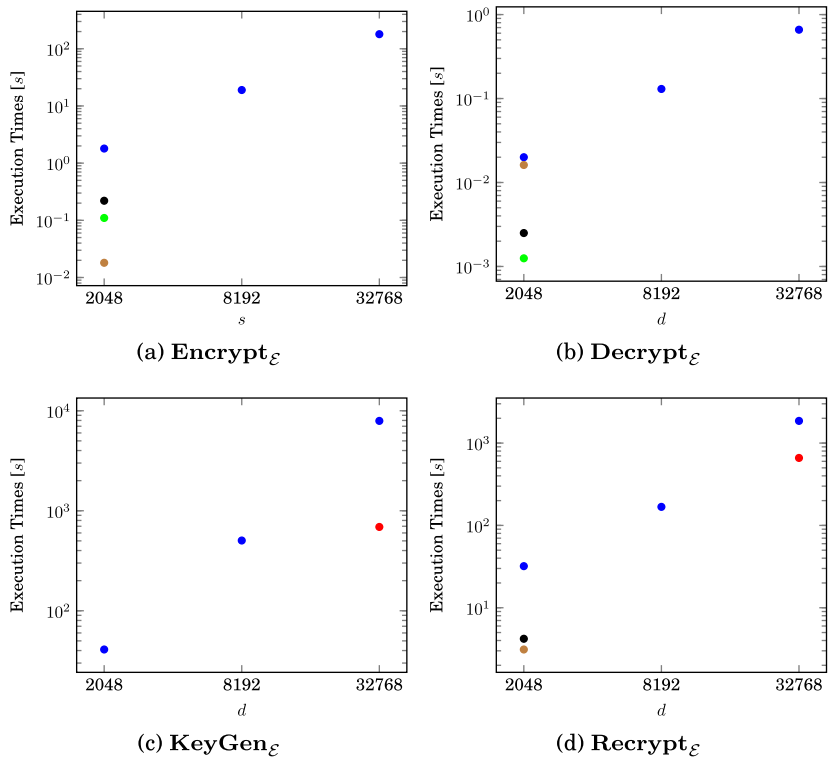


Fig. 5. Performance figures of Gentry's scheme (log-log scale): ● Implementation on an Intel Xeon E5450 (Gentry and Halevi 2011). ● Implementation on an NVIDIA Tesla C2050 (Wang et al. 2012). ● Implementation on a Stratix-V (Wang and Huang 2013). ● Implementation on a 12×AMD Opteron 6168 (Dalibard 2011). ● ASIC Implementation with a 90nm TSMC Library (Doröz et al. 2015).

Table 1. Benchmark Performance of Dalibard's Implementation (Dalibard 2011) on an Intel Core 2 Duo E6600

	Bit Representation	Large Message Space
Fibonacci	10 min 58 s	<0.01 s
$\prod_{i=k}^{k+4} i$	16 min 27 s	2 min 44 s

parallelization works by distributing the encryption of several messages over different nodes. It should be noted that increasing the throughput of encryptions results in a faster key generation procedure, since a large amount of time of the latter process corresponds to encrypting the bits of the secret key. Recryption, on the other hand, is parallelized by distributing the homomorphic evaluation of an optimized version of Equation (6), since this operation is highly parallelizable.

Further, Dalibard (2011) has shown how to change the plaintext space from $\mathbb{Z}/(2\mathbb{Z})$ to $\mathbb{Z}/(p\mathbb{Z})$, for some $p \in \mathbb{Z}$. In order to show that the larger message space was beneficial to increasing the performance of homomorphic additions and multiplications, Dalibard tested the homomorphic evaluation of two arithmetic-based benchmarks, namely, the computation of the 12 first Fibonacci numbers and the evaluation of $\prod_{i=k}^{k+4} i$. The performance results, replicated in Table 1 for $p = 16$, show a clear improvement in performance when using larger message spaces. In particular, when performing additions, there is a reduced need to do recryptions, which drastically improves performance.

Cryptosystem 4: Brakerski, Gentry and Vaikuntanathan's Scheme

We present here Brakerski, Gentry and Vaikuntanathan's Scheme [Brakerski et al. 2012] in more detail. The major contribution of this scheme is an innovative noise management technique that is itself based on a related technique from [Brakerski and Vaikuntanathan 2011b]. The underlying SHE operates as follows. Let $R_q = \mathbb{Z}_q[t]/(\Phi_m(t))$. To get the secret key, s is drawn from χ (corresponding to the error distribution), and one produces $\vec{s} = (1, s) \in R_q^2$. To get the public-key, first $\vec{x}$ is sampled uniformly from R_q^n, and $\vec{e} \leftarrow \chi^n$, where n is a parameter of the scheme used to assure security. The public-key corresponds to $Y = (-\vec{x}^T s + 2\vec{e}^T | \vec{x}^T)$. We note that $\vec{s} \times Y^T = 2\vec{e}$.

In order to encrypt a message, we represent it as $m \in R_2$ and expand the message to $\vec{m} = (m, 0) \in R_q^2$. Afterwards, we randomly select $\vec{u} \in R_2^n$ and compute $\vec{c} = \vec{m} + \vec{u} \times Y \in R_q^2$. The decryption procedure works by computing $m = [\langle \vec{c}, \vec{s} \rangle]_q \bmod 2$. The scheme is both homomorphically additive and multiplicative. However, homomorphic multiplication is computed with $\vec{c_1} \otimes \vec{c_2}$ that can be thought of as an encryption of $m_1 m_2$ under $\vec{s} \otimes \vec{s}$. In order to prevent the continuous growth of the degree of the secret-key key-switching has to be applied.

The key-switching procedure consists in manipulating the ciphertext, so that the resulting ciphertext is decryptable under a different key. To characterize key-switching, we introduce two functions. First, we define $\text{Decomp}_2(\vec{x})$ which decomposes $\vec{x} \in R_q^n$ into its bit representation $\vec{x'} \in R_2^{n\lceil \log q \rceil}$. This is performed by first writing $\vec{x} = \sum_{i=0}^{\lfloor \log q \rfloor} 2^i \vec{x'}_i$, with all $\vec{x'}_i \in R_2^n$, and then outputting $(\vec{x'}_0, \vec{x'}_1, \ldots, \vec{x'}_{\lfloor \log q \rfloor})$. The second function is $\text{Powers}_2(\vec{x})$ which expands $\vec{x} \in R_q^n$ into $(\vec{x}, 2\vec{x}, \ldots, 2^{\lfloor \log q \rfloor} \vec{x}) \in R_q^{n\lceil \log q \rceil}$. Finally, we note that $[\langle \text{Decomp}_2(\vec{c}), \text{Powers}_2(\vec{s}) \rangle]_q = [\langle \vec{c}, \vec{s} \rangle]_q$.

In order to perform key-switching, the public-key needs to be extended. Suppose we have a secret-key $\vec{s_1} \in R_q^4$, and we want to generate the parameters to change a ciphertext under this key to another key $\vec{s_2} \in R_q^2$, with parameter $n_2 = 2\lceil \log q \rceil$, and Y_2 generated as previously for the public-key. We set $Z_1 = (\text{Powers}_2(\vec{s_1})^T | 0)$, the matrix with the first column containing $\text{Powers}_2(\vec{s_1})$ is augmented with an all-zero column. We then set $\Delta_{\vec{s_1} \to \vec{s_2}} = Y_2 + Z_1$. The key-switching procedure can now be defined as:

$$\vec{c_2} = \textbf{KeySwitch}_{\mathcal{E}}(\Delta_{\vec{s_1} \to \vec{s_2}}, \vec{c_1}) = \text{Decomp}_2(\vec{c_1}) \times \Delta_{\vec{s_1} \to \vec{s_2}} \tag{7}$$

The output $\vec{c_2}$ satisfies $\langle \vec{c_2}, \vec{s_2} \rangle = 2\langle \text{Decomp}_2(\vec{c_1}), \vec{e_2} \rangle + \langle \vec{c_1}, \vec{s_1} \rangle \bmod q$, where $\vec{s_2} \times Y_2^T = 2\vec{e_2}$. We note that the use of the Decomp_2 function ensures that the operation does not introduce a significant amount of noise, since $\text{Decomp}_2(\vec{c_1})$ has only coefficients 0 or 1.

In turn, the modulus-switching technique changes the ciphertext $\vec{c} \in R_q^2$ to a ciphertext $\vec{c'} \in R_{q'}^2$, such that decrypting $\vec{c'}$ still gives the same result, and the noise is reduced by a factor of roughly $\frac{q}{q'}$. This operation consists in finding the vector $\vec{c'}$ that is the closest (using the l_1-norm) to $(q'/q)\vec{c}$ such that $\vec{c} = \vec{c'} (\text{mod } 2)$, and works for $|[\langle c, s \rangle]_q| < q/2 - (q/q')l_1(\vec{s})$. With these two techniques, one gets a leveled FHE scheme. Nevertheless, should one need to perform computations on arbitrary circuits whose depth is not known beforehand, bootstrapping still needs to be applied.

6.2 LWE-Based FHE

Regev presented the first public-key cryptosystem supported on the LWE assumption (Regev 2009). This scheme has very simple and efficient algorithms. Brakerski and Vaikuntanathan used Gentry's technique of bootstrapping to develop LWE and RLWE FHE cryptosystems (Brakerski and Vaikuntanathan 2011a, 2011b). The authors introduced a key-switching technique that decreases the size of the ciphertexts and also showed how to change the underlying modulus of the ring. With these two techniques, which shall be soon explained, it is no longer needed to "squash" the decryption circuit as previously—they reduce the dimension of the ciphertext and reduce the magnitude of the integer modulus, so that the original SHE scheme can homomorphically evaluate the degree of the new decryption algorithm.

Brakerski, Gentry, and Vaikuntanathan proposed an exceptional new FHE framework (Brakerski et al. 2012), which is described in Cryptosystem 4. The new scheme drew upon Brakerski and Vaikuntanathan's techniques of key and modulus switching to provide better noise management. With this refinement, one can linearly increase the magnitude of the public key according to the depth of the circuits one wants to process homomorphically, to avoid using bootstrapping. Schemes with this property are said to be leveled FHE schemes.

With Brakerski et al. (2012), the decryption of a ciphertext $\vec{c} \in R_q^2$, with $R_q = \mathbb{Z}_q[t]/(\Phi(t))$, which encrypts $m \in R_2$, and with a secret key $s \in R_q^2$ is computed as $m = [\langle \vec{c}, \vec{s} \rangle]_q \bmod 2$. The inner product generates a value $[m + 2\langle \vec{e}, \vec{u} \rangle]_q$, and as long as the noise term $|m + 2\langle \vec{e}, \vec{u} \rangle|$ is less than $q/2$, decryption will work correctly. The scheme is homomorphically additive, since $m_1 + m_2 = [\langle \vec{c_1} + \vec{c_2}, \vec{s} \rangle]_q \bmod 2$, as long as the error does not grow out of range. We further notice that $m_1 m_2 = [\langle \vec{c_1}, \vec{s} \rangle \langle \vec{c_2}, \vec{s} \rangle]_q \bmod 2 = [\langle \vec{c_1} \otimes \vec{c_2}, \vec{s} \otimes \vec{s} \rangle]_q \bmod 2$. Thus, $\vec{c_1} \otimes \vec{c_2}$ can be thought of as an encryption of $m_1 m_2$ under $\vec{s} \otimes \vec{s}$. Key switching can be used to change the underlying key from $\vec{s} \otimes \vec{s}$ back to $\vec{s}$ to prevent the ciphertexts from growing indefinitely.

The modulus-switching technique changes the ciphertext $\vec{c} \in R_q^2$ to a ciphertext $\vec{c'} \in R_{q'}^2$, such that decrypting $\vec{c'}$ still gives the same result, and the noise is reduced by a factor of roughly $\frac{q}{q'}$. It consists of finding the vector $\vec{c'}$ that is the closest (using the l_1-norm) to $(q'/q)\vec{c}$ such that $\vec{c} = \vec{c'} (\bmod 2)$, and works as long as $|[\langle c, s \rangle]_q| < q/2 - (q/q')l_1(\vec{s})$. With this procedure, one gets (Brakerski and Vaikuntanathan 2011b).

$$[\langle c', s \rangle]_{q'} = [\langle c, s \rangle]_q \bmod 2, \quad |[\langle c', s \rangle]_q| < (q'/q)|[\langle c, s \rangle]_q| + l_1(\vec{s}). \tag{8}$$

This means that if a short secret key $\vec{s}$ is chosen with respect to the l_1-norm, and a small enough q' relative to q, the amount of noise in the ciphertext can be significantly reduced. However, the amount of operations that one can perform homomorphically is limited not by the magnitude of the noise, but by the ratio of the noise and the modulus. Thereby, at first sight, potentially increasing that ratio might not seem useful.

Surprisingly, this technique allows one to perform a larger number of homomorphic multiplications. We start by considering the case when modulus switching is not applied. When multiplying two ciphertexts with noise x, the noise increases to x^2. If we multiply the result of two multiplications, we get a noise of x^4. And after applying this procedure l times, we get a noise of x^{2^l}. If we write q as $q \approx x^k$, we can see that one can evaluate circuits of degree at most $\log k$.

Alternatively, one can use a chain of k moduli, where each modulus q_i has a magnitude about x times smaller than q_{i-1}, that is, $q_i \approx q_{i-1}/x$, where x is the initial noise of the ciphertexts, and $q_1 \approx x^k$. If one multiplies two modulo q_1 ciphertexts, the resulting ciphertext has a noise of magnitude x^2. After applying a modulus-switching operation, both the noise and the modulus are reduced by a factor of about x, becoming x and q_2, respectively. One can see that by applying the same reasoning to the following moduli in the chain, the ciphertexts have roughly a constant noise magnitude of x.

Therefore, one is now limited by k levels of multiplication instead of $\log k$ as previously. We have an exponential improvement in the number of operations one can homomorphically perform. Now, by correctly setting the cryptosystem parameters, one can avoid bootstrapping while still being able to evaluate polynomial-sized circuits. Still, bootstrapping can be used to homomorphically evaluate arbitrary circuits without specifying their degree beforehand.

More recently, Brakerski presented a scale-invariant FHE scheme (Brakerski 2012), in which the same modulus is used throughout the computation, giving a simpler leveled FHE scheme. All the previously reported schemes encrypted messages on the Least Significant Bits (LSBs), using modulo-2 arithmetic. In contrast, Brakerski has used a Most Significant Bit (MSB) encoding—that is, with a ciphertext $\vec{c} \in \mathbb{Z}_q^n$ encrypted under the modulus q and secret key $\vec{s} \in \mathbb{Z}_q^n$, we have $\langle \vec{c}, \vec{s} \rangle = \lfloor \frac{q}{2} \rfloor m + e + qY$ for an integer Y and bounded e. Brakerski considers the fractional ciphertext $\vec{\tilde{c}} = \vec{c}/q$, and thus it holds that $\langle \vec{\tilde{c}}, \vec{s} \rangle = \frac{1}{2}m + \tilde{e} + Y$. Additive homomorphism of the scheme is immediate, and multiplicative homomorphism is achieved by tensoring the ciphertexts and decrypting the result with the tensored secret key, since $\langle 2\vec{\tilde{c}}_1 \otimes \vec{\tilde{c}}_2, \vec{s} \otimes \vec{s} \rangle = 2\langle \vec{\tilde{c}}_1, \vec{s} \rangle \langle \vec{\tilde{c}}_2, \vec{s} \rangle$. As previously, one needs to apply key switching to bring back the underlying secret key to $\vec{s}$. Brakerski further shows that $2\langle \vec{\tilde{c}}_1, \vec{s} \rangle \langle \vec{\tilde{c}}_2, \vec{s} \rangle \approx \frac{1}{2}m_1 m_2 + e' + Y'$ for small e'. In particular, the cross-term $e_1 e_2$ that was responsible for squaring the noise in previous schemes is now practically insignificant, since if $e_1 < \epsilon < 1$, $e_2 < \epsilon < 1$, for a small ϵ, $e_1 e_2 < \epsilon^2 \ll \epsilon$.

Thus, multiplication does not square the noise, but instead multiplies it by a polynomial factor $\text{pol}(\lambda)$ that depends only on the security parameter. If we have ciphertexts with noise x/q, after l levels of multiplication, the noise will grow from x/q to $(x/q)\text{pol}(\lambda)^l$, which means that a value of $q \approx x\text{pol}(\lambda)^l$ may be adopted. Brakerski also used key switching in order to decrease the dimension of the ciphertexts after each multiplication back to two. In a follow-up work, Fan and Vercauteren ported the scheme to the RLWE setting, providing for a more efficient implementation (Fan and Vercauteren 2012). We briefly describe their ideas in Cryptosystem 5.

Cryptosystem 5: Fan and Vercauteren's Scheme

Fan and Vercauteren's Scheme [Fan and Vercauteren 2012] is conceptually similar to Brakerski, Gentry and Vaikuntanathan's, except that messages are encrypted in the MSB of ciphertexts. In fact, decryption is performed by computing

$$m = \left\lfloor \left\lfloor \frac{2}{q} [\langle \vec{c}, \vec{s} \rangle]_q \right\rfloor \right\rfloor_2 \tag{9}$$

The scheme is homomorphically additive, and addition of ciphertexts adds the underlying messages. For the multiplication, one defines $\vec{c}_{mult} = (c_{mult,0}, c_{mult,1}, c_{mult,2}) = \left(\left[\left\lfloor \frac{2}{q} c_{1,0} c_{2,0} \right\rceil \right]_q, \left[\left\lfloor \frac{2}{q} (c_{1,0} c_{2,1} + c_{1,1} c_{2,0}) \right\rceil \right]_q, \left[\left\lfloor \frac{2}{q} c_{1,1} c_{2,1} \right\rceil \right]_q \right)$, and $\vec{s}_{mult} = (1, s, s^2)$. Fan and Vercauteren proved that $\vec{c}_{mult}$ is decryptable under $\vec{s}_{mult}$ using (9). Afterwards, one uses the key-switching procedure to change the underlying key to $\vec{s}$. This scheme is scale invariant and benefits from slow growth of noise.

Finally, Gentry et al. (2013) presented an LWE-based scheme that removes the need for key switching and modulus switching, as well as making bootstrapping optional. In order to make homomorphic operations easier and get rid of key switching, the ciphertexts are chosen to be matrices, and plaintexts are integers. More specifically, ciphertexts $C \in \mathbb{Z}_q^{n \times n}$ are such that $\vec{s} \times C = m\vec{s} + \vec{e}$, with plaintexts $m \in \mathbb{Z}_p$ and secret keys $\vec{s} \in \mathbb{Z}_q^n$ corresponding to near eigenvalues and near

eigenvectors of C, respectively. Homomorphic additions and multiplications are performed using matrix addition and multiplication, thus avoiding key switching. The growth of the coefficients of the ciphertexts is prevented by applying a "flattening" technique that reduces the size of the coefficients in C, while maintaining the inner product with $\vec{s}$ invariant modulo q. This scheme was adapted to the RLWE setting in Khedr et al. (2014). A description of this latter scheme can be found in Cryptosystem 6.

Cryptosystem 6: Khedr, Gulak and Vaikuntanathan's Scheme

Gentry, Sahai and Wagner [Gentry et al. 2013] showed how to avoid key and modulus switching, but at the cost of producing ciphertexts as matrices, which arguably degrades efficiency, when compared with other LWE-based schemes. Khedr, Gulak and Vaikuntanathan [Khedr et al. 2014] ported the scheme to the RLWE setting in order to improve its performance. Here, this scheme is discussed in more detail. The secret-key is a vector of polynomials $\vec{s} = (1, s)$, where $s \leftarrow \chi$, as in Brakerski, Gentry and Vaikuntanathan's scheme, and the matching public-key is a matrix $Y = (-xs + e|x)$, where $x \leftarrow R_q$. We have that $\vec{s} \times Y^T = e$. A polynomial $m \in R_2$ is encrypted as $C \in R_q^{2\times N}$, with $N = 2l$, $l = \lceil \log q \rceil$:

$$C = m\text{Powers}_2(I) + Y^T \times \vec{u} + E \tag{10}$$

where $\vec{u} \in R_q^N$ is a vector of N polynomials with random coefficients in $\{0, 1\}$, I is an $N \times N$ identity matrix and $E \leftarrow \chi^{2\times N}$. Finally, the function Powers_2 from the previous section is extended to matrices in the following way: $\text{Powers}_2 : R_q^{rl\times s} \rightarrow R_q^{r\times s}$, $\text{Powers}_2(X) = A \times X$, where $A \in R_q^{r\times rl}$ is defined as

$$A = \begin{pmatrix} 1 & 2 & \dots & 2^{l-1} & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 2 & \dots & 2^{l-1} & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 1 & 2 & \dots & 2^{l-1} \end{pmatrix} \tag{11}$$

In contrast, the Decomp_2 function is expanded such that for an input matrix $r \times s$, a matrix $rl \times s$ is outputted, where each consecutive l elements along a column contains the bit representation of each coefficient of each of the input polynomials. Decryption relies on the fact that $\vec{s} \times C = m\vec{s} \times \text{Powers}_2(I) + error$, for a "small" error. One can then obtain m by noting that the first l coefficients in the first term of this expression are in the form $m, 2m, \dots, 2^{l-1}m$, and therefore the element at location i is in the form $2^i m + error$.

One can add matrices to add the underlying messages. Homomorphic multiplication of C_1 encrypting m_1 and C_2 encrypting m_2, on the other hand is performed by computing $C_1 \times \text{Decomp}_2(C_2)$. Since $\text{Powers}_2(I) \times \text{Decomp}_2(C_2) = C_2 \times I$, it can be proved that

$$\vec{s} \times C_1 \times \text{Decomp}_2(C_2) = m_1 m_2 \vec{s} \times \text{Powers}_2(I) + error \tag{12}$$

for a small "error". Interestingly, the noise growth is not symmetric, and therefore one needs to have that in consideration when describing circuits to be evaluated. When $R_q = \mathbb{Z}_q[t]/(t^d + 1)$, multiplication of two ciphertexts C_1 and C_2 with errors bounded by b_1 and b_2 increases the error to $O(b_2 l_1(m_1) + b_1 d \log q)$ in the worst case. Therefore, when multiplying several ciphertexts it is best to do so linearly, selecting C_2 as the product accumulator, rather than building a tree of multiplications.

Batching. In many of the schemes presented in this article, the plaintext space is of the same form and has various algebraic properties. In particular, most schemes can be concretized with the following plaintext space:

$$\mathcal{M} = \mathbb{F}_p[t]/\Phi_m(t), \tag{13}$$

where p is a prime integer and $\Phi_m(t)$ is a cyclotomic polynomial of degree d. We note that for the scheme in Section 6.1, m was chosen to be a power of two, and in that case $\Phi_m(t) = t^d + 1$ with $d = \frac{m}{2}$. Batching is a technique wherein multiple messages are encrypted with a single ciphertext, exploiting the aforementioned algebraic properties, so that they can be processed in parallel. Although batching was first analyzed by Smart and Vercauteren (2009), herein we focus on how it was applied in the work by Brakerski et al. (2012). In order to deploy bootstrapping, the plaintext space of Cryptosystem 4 is modified. A prime $p = 1 (\text{mod } m)$ is selected first, and the original message space R_2 is changed to R_p. Further, homomorphic operations are realized with mod-p gates instead of Boolean ones. Finally, the modulus switching procedure is also modified such that the target q' and the original q moduli are congruent with 1 modulo p, and the modified ciphertext $\vec{c'}$ is the closest to $(q'/q)\vec{c}$ satisfying $\vec{c'} = \vec{c} (\text{mod } p)$.

We note that for a prime $p = 1 (\text{mod } m)$, (p) can be written as a product of ideals in $R = \mathbb{Z}[t]/\Phi_m(t)$. In particular, $(p) = \prod_{i=0}^{d-1} y_i$, where the y_i are ideals of the form $(p, t - \zeta_i)$, and the ζ_i are m^{th} primitive roots of unity modulo p. That is, all y_i-elements can be expressed as $u_1 p + u_2(t - \zeta_i)$ for some $u_1, u_2 \in R$. In this setting, one can apply the Chinese Remainder Theorem (CRT) since these ideals are pairwise coprime:

$$R_p \cong R_{y_0} \times \cdots \times R_{y_{d-1}}. \tag{14}$$

Consequently, d independent mod-p plaintexts can be encoded as a unique element in R_p; that is, in each ciphertext there are d unrelated message "slots." We can build automorphisms over R_p, $\tau_{i \to j}$, that permute slots i and j. Specifically, $\tau_{i \to j}$ takes R-elements $x = x(t) = x_{d-1}t^{d-1} + \cdots + x_1 t + x_0$ and maps them to $x(t^{e_{ij}}) = x_{d-1}t^{e_{ij}(d-1)} + \ldots x_1 t^{e_{ij}} + x_0$, where e_{ij} is an integer in $\mathbb{Z}_m^*$. In order to compress and decompress multiple ciphertexts into a single one, procedures $\mathbf{Pack}_{\mathcal{E}}$ and $\mathbf{Unpack}_{\mathcal{E}}$ are used, respectively, exploiting not only automorphisms but also key switching.

In a follow-up work (Gentry et al. 2011), Gentry, Halevi, and Smart have further developed the concept of batching in order to minimize the use of the inefficient $\mathbf{Pack}_{\mathcal{E}}$ and $\mathbf{Unpack}_{\mathcal{E}}$ operations. Whereas the previously described batching method can be efficient when evaluating the same function multiple times in parallel, it is not as useful when processing a complex circuit once. Therefore, an optimized technique of permuting ciphertext slots was introduced. It should be noted that it is possible to relax the condition that $p = 1 (\text{mod } m)$, but that possibly fewer plaintext slots will be available.

Finally, batching has applications beyond improving the performance of homomorphic operations. In Kaptchuk et al. (2017), a heuristic is used to encode multiple data in a ciphertext that provides integrity checking. When asked to decipher data resulting from the homomorphic evaluation of a circuit, a transcript of the applied computation is also provided. A verification method is used to ensure that the ciphertexts correspond in fact to the evaluation of the correct circuit.

6.2.1 NTRU-Based FHE. The NTRU cryptosystem (Hoffstein et al. 1998) is an LBC lacking provable security. Sthelé and Steinfeld altered this scheme so that breaking it was as hard as solving standard lattice problems, such as the SVP (Stehlé and Steinfeld 2011). This scheme was further developed in López-Alt et al. (2012) to produce a FHE scheme. Notwithstanding, a nonstandard security assumption was made so as to enable homomorphic operations while still proving security, namely, the Decisional Small Polynomial Ratio (DSPR). We note that Sthelé and Steinfeld's scheme is already somewhat homomorphic, and López-Alt, Tromer, and Vaikuntanathan showed

how to employ key and modulus switching, thus achieving a leveled FHE scheme, or alternatively a bootstrappable scheme. Bos et al. later proposed a new FHE scheme (Bos et al. 2013), also supported on Stehlé and Steinfeld's scheme, that provided security solely based on standard lattice assumptions. It used a scale-invariant perspective so as to remove the need for modulus switching. Further, in order to improve the efficiency of the scheme, a variant was proposed therein but underpinned by the DSPR assumption. Finally, Doröz and Sunar (2016) adapted the matrix-based cryptographic scheme in Gentry et al. (2013) to the NTRU setting, supporting security on standard lattice assumptions.

We do not describe these schemes here in detail, since they share many characteristics with the previously presented RLWE schemes. However, we note that when using NTRU it is possible to represent ciphertexts using a single polynomial element, which accelerates their homomorphic evaluation, but the noise growth rate increases (Lepoint and Naehrig 2014).

6.2.2 Performance. With RLWE-based schemes, it became more feasible to homomorphically evaluate complete circuits. In Figure 6, we replicate the performance of different schemes for several benchmarks. The first four circuits relate to statistical circuits, which might be useful for data-mining applications. In one application, a company would hire the computing services of another company, with the purpose of evaluating the purchasing patterns of its customers, but wanted to keep the customer details encrypted to protect their privacy. On the one hand, Naehrig et al. (2011) considered the circuits for computing the average and the standard deviation, and supported their implementation on Brakerski and Vaikuntanathan (2011b). On the other hand, Wu and Haven (2012) focused on the computation of the covariance and linear regression of datasets, and based their implementation on Fan and Vercauteren (2012).

We can see that Nehring, Lauter, and Vaikuntanathan focused on the evaluation of circuits that only require a single or few multiplications. The considered circuits were the average (avg) of n terms, $\text{avg} = \frac{\sum_{i=0}^{n-1} x_i}{n}$, which is returned as a pair $(\sum_{i=0}^{n-1} x_i, n)$, and the standard deviation (σ), $\sigma = \sqrt{\frac{\sum_{i=0}^{n-1}(x_i - \text{avg})^2}{n}}$, returned as the pair $(\sum_{i=0}^{n-1}(x_i - \text{avg})^2, n)$. The numerators and denominators are retrieved as separate encryptions since there was no way to efficiently compute divisions. In order to compute the sum of n integers, it is proposed to represent the integers' bits as coefficients of a polynomial, $px(t) = \sum_{i=0}^{d} x^{(i)} t^i$. Hence, adding the corresponding encryptions adds up the polynomials coefficient-wise. If the message space is set to $\mathbb{Z}_p$, it is possible to add up to $p - 1$ messages, and evaluating the polynomial at $t = 2$ after decryption returns the value of the sum. Regarding multiplications, the same trick is applied, but since the degree of the polynomials increases with the multiplications, integers are encoded as low-degree polynomials such that after applying multiplications, the degree is less than d (where d is the degree of the underlying ring polynomial).

Different number representations have been considered in Costache et al. (2016) and Cheon et al. (2016). In the former, approximate arithmetic is considered. Ciphertexts are formed as in Cryptosystem 4, but plaintexts are no longer computed modulo 2. Noise is no longer multiplied by 2, and is considered part of the approximate representation of numbers. When applying a rescaling procedure, analogous to modulus switching, both the noise and the plaintext are scaled down. The resulting homomorphic operations provide a more natural map to fixed-point arithmetic, enabling more efficient methods for the evaluation of transcendental functions and FFTs. In Cheon et al. (2016), it is noted that in the ring $\mathbb{Z}[t]/(t^d + 1)$, t is a formal primitive $2d^{\text{th}}$ root of unity. After approximating a value $x \in \mathbb{C}$ by $\sum_j x_j \zeta_{2d}^{j2d/N}$, where $\zeta_{2d} \in \mathbb{C}$ is a primitive $2d^{\text{th}}$ root of unity, this representation is converted to $\mathbb{Z}[t]/(t^d + 1)$ through the map $\zeta_{2d} \to t$. With this scheme, multiplications by roots of unity correspond to multiplications by powers of t, leading to efficient evaluations of N-point FFTs.

In contrast, Wu and Haven studied operations that can be concretized as matrix multiplications, which reduce to the computation of a series of inner products. Multiple optimizations were

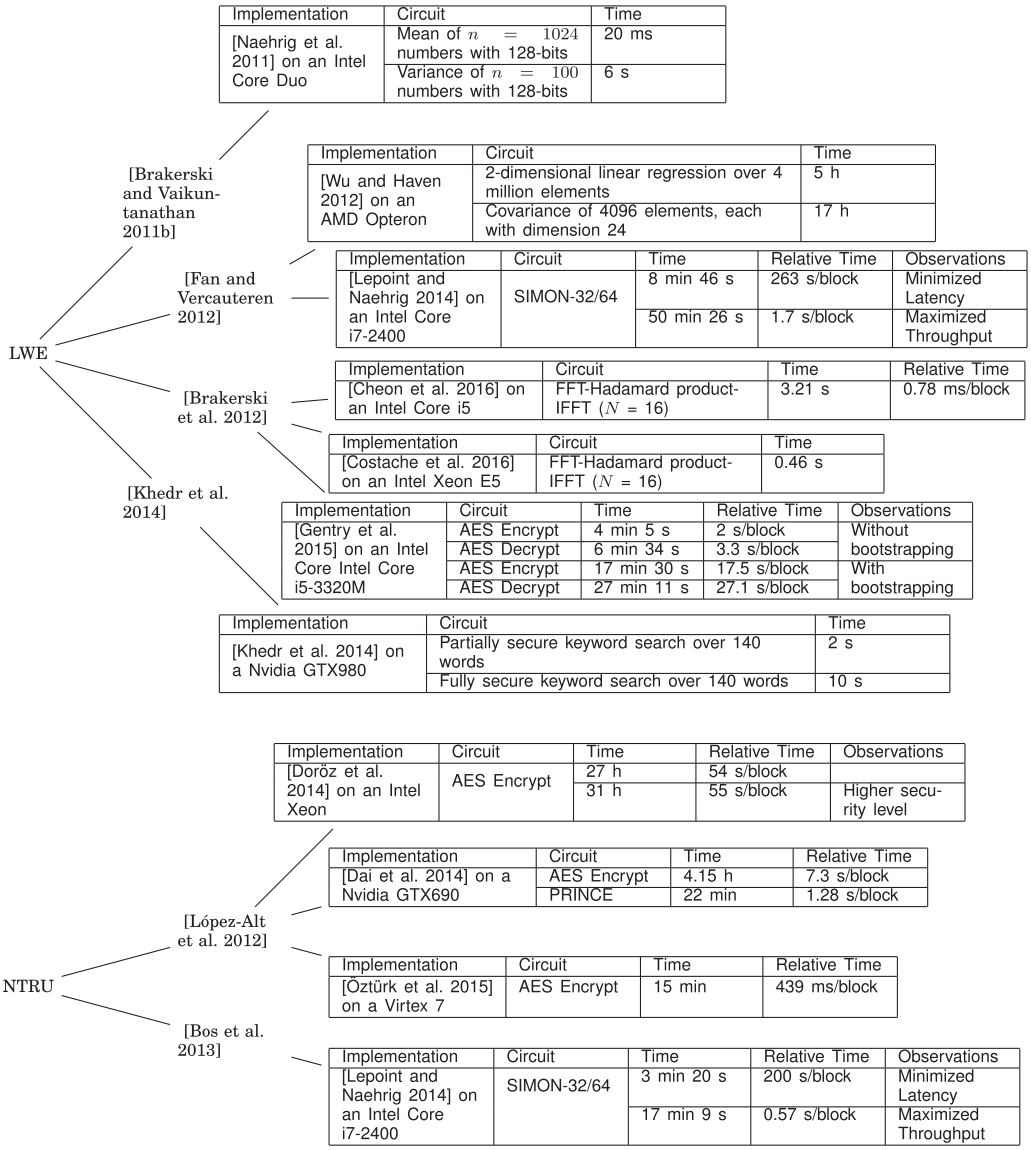


Fig. 6. Execution time of the homomorphic evaluation of several circuits. The second level of the trees corresponds to the cryptographic scheme underpinning the implementations.

introduced so as to enhance the computation of inner products. The first one regards the representation of data. Several different plaintext spaces were used, so as to leverage the CRT and perform multiprecision arithmetic. On another optimization, the batching technique was used to reduce the amount of multiplications that need to be performed. Thus, they were able to compute several multiplications of the inner product simultaneously and then use "rotations" of the plaintext space to add their value. Also, to reduce the amount of inefficient computations, key switching was only performed at the end of the calculus of the inner product, instead of after each multiplication.

A downside of fully homomorphic public-key encryption is that when messages are encrypted, the bit length of the resulting ciphertexts is several times larger than the original messages. A

client can produce ciphertexts using a "light" symmetric encryption scheme and encrypt the corresponding secret key, so as to reduce the size of ciphertexts while they are being transferred to a server. Upon receiving them, the server will homomorphically decrypt the symmetric encryption scheme ciphertext and operate on the resulting ciphertexts. With this in mind, Doröz et al. compared the homomorphic evaluation of multiple lightweight symmetric circuits (Doröz et al. 2014). This research concluded that Prince offers significant improvements over other encryption schemes such as the Advanced Encryption Standard (AES). For similar security levels, they were able to reduce roughly by half not only the polynomial degree but also the bit length of the modulus of the underlying NTRU scheme when comparing the homomorphic evaluation of the Prince and the AES circuits.

Several other implementations have considered the homomorphic evaluation of multiple symmetric-key cryptosystems. A common optimization of these implementations is that of reducing the complexity of handling large numbers using CRT remaindering and of handling large polynomials using an FFT. Gentry, Halevi, and Smart described an implementation of leveled FHE (Gentry et al. 2015) based on Brakerski et al. (2012) that exploits modulus switching, focusing on the homomorphic processing of the AES algorithm. In their implementation, the moduli $q_1, \ldots, q_l$ in the moduli chain are chosen as $q_k = \prod_{i=1}^{k} q'_i$ so that polynomials are represented using FFT coefficients modulo each prime of $\{q'_1, \ldots, q'_k\}$. In contrast, Doröz et al. (2014) developed a customized implementation of López-Alt et al. (2012) also to evaluate the AES circuit. They decreased the size of the public key by using a single private/public-key pair and choosing the moduli chain as the powers of a prime. Dai et al. (2014) later implemented this optimized scheme using CRT remaindering on GPUs. The CRT allows one to represent the large coefficients of the polynomials as many small coefficients. The FFT is afterward applied to those small coefficients to perform polynomial multiplications. The scheme was later on adapted to FPGAs (Öztürk et al. 2015) by offloading the computation of FFTs and polynomial multiplications to an FPGA. It should be noted that while FFTs are typically used to multiply polynomials, the exploitation of the Karatsuba multiplication was proposed in Migliore et al. (2017), since it enables choosing q as a power-of-two modulus and a tighter integration with relinearization.

In the case of Khedr et al. (2014), the modulus is small and therefore one does not need to employ CRT remaindering. Nevertheless, they employ the FFT to multiply polynomials, and in particular implement a version that exploits serial memory accesses and is GPU-friendly. The implementation of Khedr et al. (2014) focuses, as an application of HE, on the search of a keyword in a file. In one instance, the scheme is partially secure in that the word to be searched is provided in the clear. In contrast, in the fully secure keyword search, the keyword is provided in an encrypted form.

A line of research focusing on how fast bootstrapping could be was pursued in Ducas and Micciancio (2015) and Chillotti et al. (2016). In Ducas and Micciancio (2015), the evaluation of a NAND gate, which corresponds to a complete Boolean operation, is considered. Bits are encrypted for a plaintext modulo of 4, such that for $m_1, m_2 \in \{0, 1\}$, $m_1 + m_2 = 2$ if $\overline{m_1 \wedge m_2} = 0$ or $m_1 + m_1 \in \{0, 1\}$ otherwise. The value of $\overline{m_1 \wedge m_2}$ is produced for a plaintext modulo of 2 from $m_1 + m_2$ through an affine transformation, leading to a small noise growth. Bootstrapping is implemented by homomorphically and efficiently extracting the MSB of $b - \langle \vec{a}, \vec{s} \rangle, b \in \mathbb{Z}_q, \vec{a}, \vec{s} \in \mathbb{Z}_q^n$, with a variant of Gentry et al. (2013) in less than 1 second for an Intel core running at 3GHz. In Chillotti et al. (2016), this scheme was optimized, leading to speedups of over 10.

6.2.3 Comparison Between NTRU and RLWE. Lepoint and Naehrig (2014) conducted theoretical and experimental comparisons between the presumably most practical homomorphic encryption schemes of Fan and Vercauteren (2012), based on RLWE, and Bos et al. (2013), based on NTRU. It has been observed both in theory and practice that the noise growth rate is smaller for Fan

Cryptosystem 7: van Dijk et al.'s scheme

A ciphertext encrypting a bit m under van Dijk et al.'s scheme has the format $c = m + 2u + kq$, where u and k are random values and q is an hidden odd integer that corresponds to the private-key. It can be seen that when two ciphertexts $c_1 = m_1 + u_1 + k_1 q$ and $c_2 = m_2 + 2u_2 + k_2 q$ are added or multiplied, the result retains the format $c_3 = (m_1 + m_2) + 2u_3 + k_3 q$ or $c_4 = (m_1 m_2) + 2u_4 + k_4 q$, respectively. Therefore to produce one such ciphertext, one can add m to multiple encryptions of 0. In particular, the public-key is defined to be a tuple of $\tau+1$ elements $\{x_0, \ldots, x_\tau\}$ drawn from the following distribution:

$$\mathcal{D}_{\gamma,\rho}(q) = \{\text{choose } r \leftarrow \mathbb{Z} \cap [0, 2^\gamma/q), s \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \text{ output } x = qr + s\} \quad (15)$$

where γ, ρ and the bit-length of q are parameters chosen to ensure the security and correctness of the scheme. Additionally, the tuple is generated on the condition that x_0 is odd, that it is the largest sample and that $x_0 \bmod q$ is even. By noticing that $2u$, for a random $u \leftarrow (-2^{\rho'}, 2^{\rho'})$, and $x'_i = [2x_i]_{x_0}$ are encryptions of 0, one obtains the following encryption procedure:

$$c = \left[m + 2u + 2 \sum_{i \in S} x_i \right]_{x_0} \quad (16)$$

for a random subset $S \subseteq \{1, \ldots, \tau\}$.

As previously noted, addition and multiplication of ciphertexts can be used to effectively implement the logical XOR and AND operations of the encrypted bits, respectively. Whereas the bit-length of freshly encrypted ciphertexts is limited by the size of x_0, their size grows as numbers are added and multiplied. While, at first sight, it might seem that reducing the result modulo x_0 would provide a solution, this operation would lead to subtracting a large multiple of x_0 after each multiplication, and the result would no longer be small modulo q. One efficient solution to this problem is to choose x_0 to be an exact multiple of q, enabling one to reduce the results of multiplications modulo x_0 without changing the result modulo q. However, one now needs to further assume the hardness of factoring large integers to prove security.

Another downside of this cryptosystem is that the degree of the permitted circuits is limited due to the noise growth. van Dijk et al. follow Gentry's approach (cf. Section 6.1) of homomorphically evaluating the decryption circuit so as to "refresh" the ciphertexts and enable further homomorphic operations. A ciphertext can be decrypted with $m = (c \bmod q) \bmod 2$, which simplifies to $m = [c - \lfloor c/q \rceil q]_2 = (c \bmod 2) \oplus (\lfloor c/q \rceil \bmod 2)$, since q is odd. No way was found for homomorphically computing the deciphering circuit directly, thus they "squash" the decryption circuit to reduce the circuit degree.

With this purpose, a SSSP instance is added to the public-key, viz. a large set of rational numbers $\mathcal{S} = \{x_i | i \in \{0, \ldots, S-1\}$ is published, and a subset of these adds to $\approx 1/q (\bmod 2)$. Moreover, a vector of encryptions of $\vec{\delta} = (\delta_0, \ldots, \delta_{S-1})$ is published where $\vec{\delta}$ satisfies $\frac{1}{q} \approx \sum_{i=0}^{S-1} \delta_i \times x_i (\bmod 2)$. Subsequently, in order to decipher a ciphertext c, one computes $\vec{y} = (y_0, \ldots, y_{S-1})$, where $y_i = [cx_i]_2$, leaving $\lceil \log S \rceil + 3$ bits after the binary point, and finally setting $m = (c \bmod 2) \oplus (\lfloor \sum_i \delta_i y_i \rceil \bmod 2)$. Now, it is possible to homomorphically evaluate the decryption circuit and therefore achieve a FHE scheme.

and Vercauteren's scheme than for Bos et al.'s. Conversely, due to the ciphertext structure (we recall that Fan and Vercauteren's ciphertexts consist of two polynomial ring elements, whereas Bos et al.'s consist of one), it takes twice more time to encrypt or add ciphertexts using Fan and Vercauteren's scheme, and three times longer to multiply two ciphertexts, when compared with Bos et al.'s scheme.

6.3 Approximate Greatest Common Divisor-Based FHE

A new conceptually simple FHE scheme was proposed by van Dijk et al. (2010), which is described in Cryptosystem 7. The blueprint of Gentry was followed. An SHE scheme was proposed, and the decryption circuit was "squashed" so that its circuit fitted into the homomorphic capacity of the SHE scheme. Since bootstrappability is achieved, it is possible to homomorphically evaluate arbitrary circuits. The simplicity of the scheme derives from the fact that it is supported on the AGCD problem.

Definition 6 (AGCD). Given $pol(\lambda)$ (λ is the security parameter) samples $x_i = qr_i + s_i$, where $r_i \leftarrow \mathbb{Z} \cap [0, 2^\gamma/q)$, $s_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$, and $\gamma, \rho \in \mathbb{Z}$ for an η-bit odd integer q, find q.

In fact, the public key is an instance of that problem, where the integer q is the secret key. In contrast to the previously described schemes, where homomorphic operations are implemented with operations on rings, herein only integer operations have to performed. Nevertheless, the scheme is impractical insofar as the public key needs to have size $O(\lambda^{10})$, where λ is the security parameter, in order to ensure security.

Coron et al. were able to reduce the size of the public key in Coron et al. (2011). Their proposal is based on the fact that if two x_i are multiplied, for certain parameters, the result is still small modulo q. Therefore, one can store a smaller amount of x_i in the public key ($O(\lambda^7)$ instead of $O(\lambda^{10})$) and combine them multiplicatively. Ciphertexts are now computed as

$$c = \left[m + 2u + 2 \sum_{1 \le i,j \le \beta} b_{i,j} x_{i,0} x_{j,1} \right]_{x_0}, \tag{17}$$

where $x_{i,b}$ are elements of the public key. Whereas the original cryptosystem produced ciphertexts that were linear in x_i, ciphertexts are now quadratic in $x_{i,b}$. The proof of security for this scheme is based on a stronger assumption, that is, that a variant of AGCD is hard. Coron et al. (2012) later generalized this technique, employing higher degrees of the public-key elements.

A different way to compress the public key (Coron et al. 2012) corresponds to integrating in the public key a description of a pseudo-random generator. If this generator produces integers with a bit length similar to that of x_i, one only needs to further provide in the public key small corrections δ_i to these pseudo-random values χ_i, such that $x_i = \chi_i - \delta_i$ is small modulo q. This technique is fully compatible with the scheme in Coron et al. (2011), but whereas before with the set of parameters from Coron et al. (2011) one gets a public key of size 802MB, one now gets a public key size of 4.6MB.

Coron et al. (2012) also showed how to apply the modulus-switching technique to the scheme, in a similar way to that of the RLWE framework of Brakersky, Gentry, and Vaikuntanathan (cf. Cryptosystem 4). This technique enables the homomorphic evaluation of arbitrary circuits without using bootstrapping, by setting the scheme parameters appropriately. In the case of RLWE-based cryptosystems, this operation consists of multiplying a ciphertext c modulo q by $\frac{q'}{q}$ and rounding suitably, so as to produce a ciphertext c' modulo q' with a noise reduction of about $\frac{q}{q'}$. However, the same cannot be applied directly to the scheme of van Dijk et al., since both q and q' must remain secret. This problem was solved through the introduction of two changes (Coron

et al. 2012). First, the value of $2^{\eta'}/q$ was provided in the public key, but hidden as an instance of an SSSP. Second, an indication vector associated with the subset of the SSSP is encrypted with the new private key q'. When performing modulus switching, the modulus q is replaced by $2^{\eta'}$, while "hidden" in the SSSP. Then, the encrypted indication vector is used, such that when c' is computed, the ciphertext encrypts the same value as c except that the underlying key is now q'. This scheme is described in more detail in Cryptosystem 8.

Cryptosystem 8: Coron, Naccache and Tibouchi's Scheme

The modulus-switching technique consists in changing the underlying secret-key from q to q', such that there is roughly a decrease of noise by a factor of q/q', and was first applied to AGCD cryptosystems in [Coron et al. 2012]. To do so, one first produces a "virtual ciphertext" with underlying modulus $2^{\eta'}$. The public-key now incorporates a vector $\vec{x} = \{x_0, \ldots, x_{S-1}\}$ of rational numbers, that is an instance of a SSSP. Moreover, the vector $\vec{\delta'} = \text{Powers}_2(\vec{\delta}) = (\vec{\delta}, 2\vec{\delta}, \ldots, 2^{\eta'}\vec{\delta})$ is defined, where $\vec{\delta} = (\delta_0, \ldots, \delta_{S-1})$ is a vector of bits such that $\frac{2^{\eta'}}{q} \approx \sum_{i=0}^{S-1} x_i \times \delta_i \bmod 2^{\eta'+1}$. The public-key is further expanded with a vector encryption $\vec{c}_{\vec{\delta'}}$ of $\vec{\delta'}$ under q', computed as follows:

$$\vec{c}_{\vec{\delta'}} = q'\vec{r} + \vec{s} + \left\lfloor \frac{\vec{\delta'} q'}{2^{\eta'+1}} \right\rceil \tag{18}$$

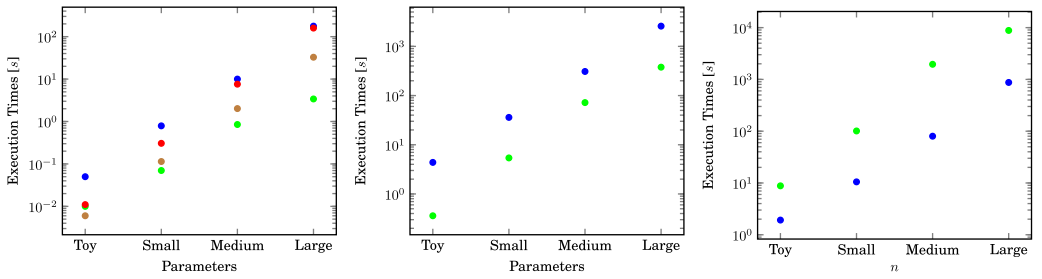
where $\vec{r}$ and $\vec{s}$ are random integer vectors, whose bit-length needs to be carefully set so as to ensure both security and correctness of the scheme.
The new "virtual ciphertext" is hidden in a SSSP instance under q' by determining $\vec{c^*} = (\lfloor cx_i \rceil \bmod 2^{\eta'+1})_{0 \le i \le S-1}$. Afterwards, $\vec{c^{**}} = \text{Decomp}_2(\vec{c^*})$ is computed (the decomposition of $\vec{c^*}$ to $\eta'+1$ vectors with its binary digits), and this result is collapsed into $c' = 2\langle \vec{c}_{\vec{\delta'}}, \vec{c^{**}} \rangle + [c]_2$. With this technique, it is no longer needed to apply recryption to evaluate arbitrary circuits, since it is possible to build leveled AGCD-based FHE schemes.

Coron, Lepoint, and Tibouchi ported the scale-invariant perspective of Fan and Vercauteren's scheme (cf. Cryptosystem 5) to the integer-based homomorphic scheme (Coron et al. 2014), as described in Cryptosystem 9. A bit m is now encoded as 1 when c is "close" to a multiple of $q/2$, and as 0 otherwise. Furthermore, ciphertexts operate with respect to modulo q^2 instead of q. This enables a reduction in the noise growth. Unfortunately, this method does not completely remove the need for modulus switching, since after each multiplication the message that was encrypted as the MSB of $c \bmod q$ becomes encrypted as the MSB of $c \bmod q^2$. Thus, a modulus-switching operation has to be applied to bring q^2 back down to q.

Furthermore, Kim et al. and Coron, Lepoint, and Tibouchi independently generalized the scheme of van Dijk et al., using the CRT, to support multiple pairwise coprime integers as the secret key (Kim et al. 2013; Coron et al. 2013). The scheme has a relatively small overhead, but enables Single Instruction Multiple Data (SIMD)-style operations or large message spaces. The novel ciphertexts have the form

$$c = \text{CRT}_{q_0,\ldots,q_{l-1}}(u_0, m_1 + u_1 p_1, \ldots, m_{l-1} + u_{l-1} p_{l-1}), \tag{21}$$

where the secret key corresponds to a set of pairwise coprime integers q_i, for $i \in \{0, \ldots, l-1\}$. The message space is $\mathbb{Z}_p$ for $p = \prod_{i=1}^{l-1} p_i$, and $m_i = m \bmod p_i$. The value of m is decrypted by computing the value $m = \text{CRT}_{p_1,\ldots,p_{l-1}}(d_1, \ldots, d_{l-1})$, with $d_i = [c \bmod q_i]_{p_i}$, for all $i > 0$.



(a) $\mathbf{Encrypt}_{\mathcal{E}}$ (b) $\mathbf{KeyGen}_{\mathcal{E}}$ (c) $\mathbf{Recrypt}_{\mathcal{E}}$

Fig. 7. Performance figures of AGCD scheme (y-axis log scale): ● Implementation on an Intel Core2 Duo E8500 (Coron et al. 2011). ● Implementation on an Intel Core2 Duo E8400 (Coron et al. 2012). ● Implementation on a Virtex-7 (Cao et al. 2013). ● Implementation on a Virtex-7 (Moore et al. 2014).

Cryptosystem 9: Coron, Lepoint and Tibouchi's Scheme

To get a scale-invariant scheme [Coron et al. 2014], one should use the MSB of $c \bmod q$ to encode the value of a bit m. Moreover, one should operate modulo q^2 instead of q. In concrete, the ciphertext c of the message m has the following format:

$$c = u + (m + 2u^*)\frac{q-1}{2} + sq^2 \tag{19}$$

where u, u^* and s are random integers. Decryption is performed by computing $m = [2c \bmod q]_2$. The scheme is clearly homomorphic additive. However, when two ciphertexts are multiplied, the result is obtained as the MSB of $c \bmod q^2$. When multiplying two ciphertexts c_1 and c_2 encrypting bits m_1 and m_2 respectively, one gets c_3:

$$c_3 = 2c_1c_2 = u + (m_1m_2)\frac{q^2-1}{2} + sq^2, \tag{20}$$

where $u > q$ but still $u \ll q^2$. Afterwards, the modulus switching technique discussed in Cryptosystem 8 has to be applied in order to bring back the underlying modulus from q^2 to q. As a result, ciphertexts noise increases only linearly with the amount of homomorphic multiplications.

The described scheme is both additive and multiplicative homomorphic, given that the u_i remain sufficiently shorter than q_i, and the p_i are sufficiently small. Cheon et al. later showed how to implement permutations of the underlying SIMD slots (Cheon et al. 2013). Unlike Brakerski, Gentry, and Vaikuntanathan's scheme (cf. Section 6.2), there is no algebraic structure to exploit; therefore, the permutation of elements is embedded into the process of bootstrapping.

6.3.1 Performance. We compare in Figure 7 the performance of the implementations in Coron et al. (2011) and Coron et al. (2012). We do not report the execution times for decryption since they are negligible when compared with the other operations. The first scheme used a host of optimizations to decrease both the size of the keys and the deciphering circuit, some of which are based on Gentry and Halevi (2011). In contrast, the second scheme is based on modulus switching.

Table 2. Execution Time of the Homomorphic Processing of AES Ciphering for Two AGCD Schemes

AGCD					
Implementation & Scheme	Parameters	Circuit	Platform	Execution Time	Relative Time
Coron et al. (2013)	Toy	AES Encryption	Intel Core i7	0.08 h	29 s/block
	Small			0.74 h	1 min 12 s/block
	Medium			7.86 h	3 min 25 s/block
	Large			113 h	12 min 46 s/block
Coron et al. (2014)	Toy	AES Encryption	Intel Xeon E5-2690	15.1 s	1.7 s/block
	Small			1 min 40 s	2.9 s/block
	Medium			13 min 29 s	5.8 s/block
	Large			3 h 35 min	23 s/block

We note that even though the recryption operation is significantly slower for the latter than in the former scheme, this operation is no longer strictly necessary.

FPGA accelerators to the encryption procedure were considered in Cao et al. (2013) and Moore et al. (2014). Both architectures focus on the multiplication of large integers, but while the former uses an FFT approach, the latter exploits the Comba algorithm. It is suggested that the Comba algorithm is more effective for when the bit length of the two operands differs largely. Furthermore, it enables a better use of the FPGA resources.

The performance of homomorphically performing AES ciphering reported in Coron et al. (2013) and Coron et al. (2014) is replicated in Table 2. The first scheme, implemented by Coron, Lepoint, and Tibouchi, used batching and applied the ciphertext compression technique in Coron et al. (2012). In contrast, the second scheme, also implemented by Coron, Lepoint, and Tibouchi, exploited scale invariance. Using this scheme, large parameters were chosen so as to avoid recryption, yielding a significant improvement in performance when compared to the previous one.

7 CONSIDERATIONS ON THE SECURITY OF SHE/FHE SCHEMES

The goal of this section is not to exhaustively describe all the existent attacks to SHE/FHE schemes, but rather to identify the types of attacks to SHE/FHE schemes that attracted considerable attention over the past years. In particular, we identify efforts devoted to the fundamental problems underpinning the security of classical schemes, like RSA and El Gamal; prominent postquantum schemes, such as those based on lattices; and others. Following implicit definitions given in Katz and Lindell (2007), we separate these between *passive* and *active* attacks.

7.1 Passive Attacks

In this section, we briefly walk through attacks to RSA and El Gamal, in Section 7.1.1, and LBCs, in Section 7.1.2.

7.1.1 RSA and El Gamal. The security of RSA relies on the hardness of factoring large integers. The exact relation to factoring is described in Menezes et al. (1996) (Section 8.2.2). By factoring the modulus q of an RSA public key (q, e), an attacker can easily find the decryption exponent d. This is usually referred to as a brute-force attack on RSA (Boneh 1999). The most efficient algorithm to date is Pollard's General Number Field Sieve (GNFS), which is more than 20 years old. Ever since the GNFS algorithm was designed, no major developments or breakthroughs were published on integer factorization, except for polynomial-time factoring on quantum computers (cf. Shor (1994)) and polynomial-time primality proving (Bos et al. 2009). The reader can find more about integer factorization and other attacks to the RSA in Boneh (1999), Bos et al. (2009), and Kleinjung et al. (2010).

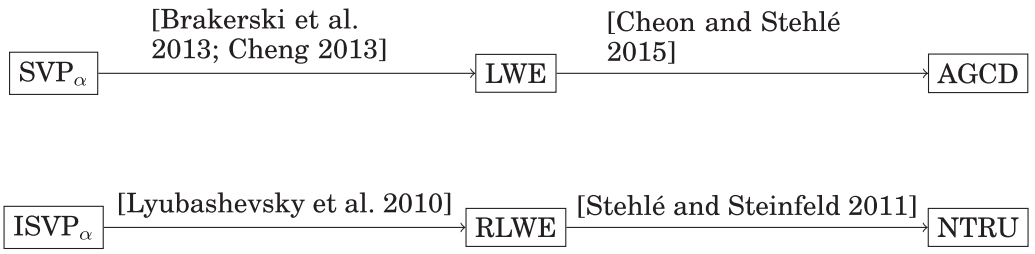


Fig. 8. Hardness reduction diagram. An arrow $A \rightarrow B$ means that A is reducible to B, that is, that B is at least as hard as A.

The security of the El Gamal public key cryptosystem is supported on the intractability of the discrete logarithm problem. Over the past three decades, a large amount of research has been dedicated to solving discrete logarithms, and multiple algorithmic advances have been proposed. Nonetheless, this remains a hard problem in many of the possible groups used in practice. For more about this problem, we refer the reader to the following comprehensive surveys (Joux et al. 2014; Menezes et al. 1996).

7.1.2 Lattice-Based Schemes. Similarly to RSA, whose security is based on factoring large numbers, and El Gamal, whose security is supported on the hardness of computing discrete logarithms, LBCs are also supported on hard mathematical problems, and in particular hard lattice problems. In this context, we can identify the CVP and the SVP as the main theoretical foundations of many LBCs. Furthermore, it has been proven that the LWE and RLWE are at least as hard to solve as approximate versions of the SVP and the Ideal Shortest Vector Problem (ISVP), respectively (Brakerski et al. 2013; Cheng 2013; Lyubashevsky et al. 2010). The ISVP is similar to the SVP, but restricted to ideal lattices. There exist also hardness reductions from the LWE and RLWE problems to the AGCD and NTRU, respectively, for specific parameters (Cheon and Stehlé 2015; Stehlé and Steinfeld 2011). Figure 8 provides an overview of this discussion.

Definition 3 provides two main instances of the GLWE problem: the LWE and the RLWE. Whereas the former system is related to problems in general lattices, the latter is connected to problems in ideal lattices. The specialization of LWE to ideal lattices resulted from a need to improve efficiency, since the additional structures provide for smaller key sizes and faster cryptographic operations. However, other approaches are possible for this specialization, such as the Polynomial Learning with Errors (PLWE) (Castryck et al. 2016), for which the underlying polynomial modulo is not restricted to be cyclotomic, and different error distributions χ are considered (namely, instead of defining a Gaussian over R, a Gaussian over the integers is sampled for each coefficient of the polynomial). It has been proven that some of these constructions are not secure (Castryck et al. 2016); namely, when it is possible to build a ring homomorphism such that the image of the error distribution is distinguishable from uniform, and the codomain is sufficiently small to be enumerated, a security risk is posed.

The added structure of working with rings has also been exploited in the context of NTRU and a variant of Gentry's scheme (cf. Section 6.1), which for wrongly chosen parameters lead to subexponential and quantum polynomial attacks on the underlying security assumptions, respectively (Albrecht et al. 2016; Cramer et al. 2016). The scheme in López-Alt et al. (2012) and the best-performant scheme in Bos et al. (2013) have been severely affected by the attack in Albrecht et al. (2016), since the secret key is chosen with an unusually small norm to enable larger homomorphic multiplicative depths. In contrast, with Doröz and Sunar (2016) and with the most secure

scheme in Bos et al. (2013), larger secret-key norms can be used and the reduction in Stehlé and Steinfeld (2011) is applicable. Thus, the attack by Albrecht et al. (2016) is prevented.

Understanding the practicability of lattice problems is crucial to determine the parameters of LBCs. Algorithms to solve these problems have been evolving at a rapid pace, which motivated the construction of an online challenge, the SVP challenge,[1] which in turn receives and publishes the highest lattice dimensions on which the SVP was solved.

Enumeration and sieving are the two biggest families of SVP solvers. Specific algorithms within these classes can be adapted to solve the CVP. Sieving and enumeration algorithms have competed for the throne among SVP solvers since 2001. While enumeration has been studied for decades, the progress in sieving algorithms took off much later, in 2008, when they were shown to be tractable for moderate dimensions. On random lattices, enumeration with extreme pruning is still more practical than any known sieving algorithm, at least for moderate dimensions. However, sieving algorithms can be considerably accelerated in practice for specific types of lattices, such as ideal lattices. Thus, both classes of algorithms are important and remain under study. Other classes of algorithms exist, but they are essentially interesting from a theoretical standpoint, and have no practical implications.

Up until this point, the highest lattice dimension reported on the SVP challenge was 144, solved with the Extended Restricting Reduction (ERR) algorithm (Fukase and Kashiwabara 2015). To reach dimension 144, the authors developed a parallel version of the ERR algorithm, which ran on 700 cores, for about 50 days, according to what has been reported on the challenge. Many mathematical problems relevant in cryptography require the use of many computing resources, with regard to both high core counts and large memories (such as factoring large integers, which we covered in Section 7.1.1), and the same happens for SVP solvers (e.g., Kuo et al. (2011)).

While in theory the SVP is the most important problem with regard to the security of LBCs, it has been argued that, in practice, it suffices to solve an approximate version of the SVP, usually referred to as SVP_α, if such a scheme is to be broken. Although there are no algorithms specifically designed to address this problem, lattice basis reduction algorithms are often used to solve it. Lattice basis reduction is the transformation of a given basis B into another basis B', which spans the same lattice but has shorter and more orthogonal vectors. With adequate parameters, the shortest vector of the basis is short enough to solve SVP_α. For instance, this problem has been solved with lattice dimensions higher than 800[2]. The most important lattice basis reduction algorithms are LLL and BKZ (Lenstra et al. 1982; Schnorr 1988), which lay the foundation of many CVP and SVP solvers and other lattice reduction algorithms. Not only are lattice basis reduction algorithms practical attacks to LBCs, but also they can be used to attack other types of schemes, ranging from RSA to those supported on the AGCD problem (Joux and Stern 1998; van Dijk et al. 2010).

7.2 Active Attacks

In contrast to the previous section, here we present attacks where an adversary needs to actively interfere with the client's system. In particular, we focus on key recovery attacks.

7.2.1 Key Recovery Attacks. Encryption schemes may enjoy ciphertext indistinguishability. In these schemes, an attacker will have a probability of ≈ 0.5 when trying to identify which of two possible plaintexts (m_0, m_1) a ciphertext c encrypts (i.e., there is no better algorithm than choosing at random). The definition of security in terms of indistinguishability depends on the assumptions that are made of the attacker. Schemes are usually considered IND-CPA, IND-CCA1, or IND-CCA2

[1] www.latticechallenge.org/svp-challenge/.
[2] www.latticechallenge.org/halloffame.php.

secure on an increasing scale of security level achieved. Clear definitions for these terms are provided in Loftus et al. (2010). Whereas the current FHE schemes cannot achieve IND-CCA1 or IND-CCA2 since one assumes an attacker can decrypt arbitrary ciphertexts and the secret key is made public in an encrypted form, the SHE schemes supporting FHE can be evaluated with respect to these definitions.

On the one hand, all the schemes used to support FHE presented in this survey are IND-CPA secure under different hardness assumptions. On the other hand, any HE scheme cannot achieve IND-CCA2 security, due to the malleability of the cryptosystems (Dahab et al. 2015). Over the past years, efforts have been devoted to determine the security degree of HE schemes, as far as IND-CCA1 is concerned. Various schemes that support exclusively either homomorphic additions or multiplications can be shown to be IND-CCA1 (e.g., Lipmaa (2008)). In contrast, the basic scheme used by Gentry and Halevi and various AGCD-, LWE-, and NTRU-based schemes have been shown to be susceptible to key recovery attacks where an adversary is capable of getting the secret key with a polynomial amount of queries to a decryption oracle (Loftus et al. 2010; Chenal and Tang 2014; Dahab et al. 2015). This type of attack is stronger than usual IND-CCA1 attacks. Fortunately, Loftus et al. (2010) have presented a scheme supporting both homomorphic additions and multiplications that achieves IND-CCA1. The scheme is a variant of an SHE scheme by Smart and Vercauteren (2009), which is similar to Gentry and Halevi's scheme (cf. Section 6.1). The authors recover the random polynomial used to encrypt the plaintext during the decryption phase and re-encrypt the result to compare against the ciphertext. This techniques consists of a form of ciphertext validity check, which supports an IND-CCA1 proof of security, through an argument stating that an adversary would not have been able to produce this ciphertext unless he or she knew what he or she was ciphering, and thus there is no further "information" that can be extracted from the decryption oracle.

8 CONCLUSION

Gentry's PhD dissertation (Gentry 2009) represented a breakthrough in the field of cryptography by presenting the first plausible scheme supporting arbitrary computations on encrypted data. This had long been considered the holy grail of cryptography, due to its utility in multiple fields, from cloud computing to data mining. The original scheme was based on ideals and lacked an efficient implementation. Nevertheless, its core idea, namely, using an SHE scheme capable of evaluating its own decryption circuit, was repeatedly applied to other cryptosystems (LWE, NTRU, and AGCD), providing for more efficient schemes. As other important techniques, such as modulus and key switching, were proposed, these systems became more and more efficient.

This survey reviews modern SHE and FHE encryption methods and techniques that derived from Gentry's work. Based on the experimental results reported for the various systems, it was shown that general FHE schemes are not yet practical from a performance perspective, due to the recryption operations. Nevertheless, schemes developed for specific applications are becoming viable, thanks to techniques such as invariant perspectives and modulus switching.

One aspect that will certainly be addressed by future work will be that of security. Even though the security levels supported by most of today's FHE schemes may be sufficient for specific applications, as was discussed in this survey, the lack of IND-CCA1 makes them unsuitable for general-purpose applications.

Finally, the discovery and development of the first FHE schemes were a major intellectual step and represent a breakthrough in the direction of solving an array of practical problems. We hope that this survey can be an important tool in the understanding of the basic concepts of SHE/FHE schemes, and in the support of further developments in this field.

REFERENCES

Ben Adida. 2015. Helios: Trust the vote. Retrieved from https://vote.heliosvoting.org/.

Martin Albrecht, Shi Bai, and Léo Ducas. 2016. *A Subfield Lattice Attack on Overstretched NTRU Assumptions*. Springer, Berlin, 153–178. DOI: https://doi.org/10.1007/978-3-662-53018-4_6

Paul Barrett. 1987. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *Advances in Cryptology (CRYPTO'86)*, Andrew M. Odlyzko (Ed.). Lecture Notes in Computer Science, Vol. 263. Springer, Berlin, 311–323. DOI: https://doi.org/10.1007/3-540-47721-7_24

Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. 2008. *Post Quantum Cryptography*. Springer Publishing Company.

Dan Boneh. 1999. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS* 46 (1999), 203–213.

Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. 2013. Private database queries using somewhat homomorphic encryption. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security (ACNS'13)*. Springer-Verlag, Berlin, 102–118. DOI: https://doi.org/10.1007/978-3-642-38980-1_7

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of the 2nd International Conference on Theory of Cryptography (TCC'05)*. Springer-Verlag, Berlin, 325–341. DOI: https://doi.org/10.1007/978-3-540-30576-7_18

Joppe W. Bos, Marcelo E. Kaihara, Thorsten Kleinjung, Arjen K. Lenstra, and Peter L. Montgomery. 2009. On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography. Cryptology ePrint Archive, Report 2009/389. Retrieved from http://eprint.iacr.org/.

Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. 2013. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, Martijn Stam (Ed.). Lecture Notes in Computer Science, Vol. 8308. Springer, Berlin, 45–64. DOI: https://doi.org/10.1007/978-3-642-45239-0_4

Zvika Brakerski. 2012. Fully Homomorphic Encryption Without Modulus Switching from Classical GapSVP. Cryptology ePrint Archive, Report 2012/078. Retrieved from http://eprint.iacr.org/.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS'12)*. ACM, New York, 309–325. DOI: https://doi.org/10.1145/2090236.2090262

Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. 2013. Classical hardness of learning with errors. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC'13)*. ACM, New York, 575–584. DOI: https://doi.org/10.1145/2488608.2488680

Zvika Brakerski and Vinod Vaikuntanathan. 2011a. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*. IEEE Computer Society, 97–106. DOI: https://doi.org/10.1109/FOCS.2011.12

Zvika Brakerski and Vinod Vaikuntanathan. 2011b. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Proceedings of the 31st Annual Conference on Advances in Cryptology (CRYPTO'11)*. Springer-Verlag, Berlin, 505–524. Retrieved from http://dl.acm.org/citation.cfm?id=2033036.2033075

Xiaolin Cao, Ciara Moore, Maire O'Neill, Elizabeth O'Sullivan, and Neil Hanley. 2013. Accelerating Fully Homomorphic Encryption over the Integers with Super-Size Hardware Multiplier and Modular Reduction. Cryptology ePrint Archive, Report 2013/616. Retrieved from http://eprint.iacr.org/2013/616.

Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. 2016. *Provably Weak Instances of Ring-LWE Revisited*. Springer, Berlin, 147–167. DOI: https://doi.org/10.1007/978-3-662-49890-3_6

Massimo Chenal and Qiang Tang. 2014. On Key Recovery Attacks Against Existing Somewhat Homomorphic Encryption Schemes. *Cryptology ePrint Archive, Report 2014/535*. Retrieved from http://eprint.iacr.org/.

Kuan Cheng. 2013. Some Complexity Results and Bit Unpredictable for Short Vector Problem. *Cryptology ePrint Archive, Report 2013/052*. Retrieved from http://eprint.iacr.org/2013/052.

Jung Hee Cheon, Jean-Sèbastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. 2013. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology (EUROCRYPT'13)*, Thomas Johanssonand Phong Q. Nguyen (Eds.). Lecture Notes in Computer Science, Vol. 7881. Springer, Berlin, 315–335. DOI: https://doi.org/10.1007/978-3-642-38348-9_20

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2016. Homomorphic Encryption for Arithmetic of Approximate Numbers. Cryptology ePrint Archive, Report 2016/421. Retrieved from http://eprint.iacr.org/2016/421.

Jung Hee Cheon and Damien Stehlé. 2015. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology (EUROCRYPT'15)*, Elisabeth Oswaldand Marc Fischlin (Eds.). Lecture Notes in Computer Science, Vol. 9056. Springer, Berlin, 513–536. DOI: https://doi.org/10.1007/978-3-662-46800-5_20

Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachne. 2016. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. Cryptology ePrint Archive, Report 2016/870. Retrieved from http://eprint.iacr.org/2016/870.

Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. 2013. Batch Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2013/036. Retrieved from http://eprint.iacr.org/.

Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. 2014. Scale-Invariant Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2014/032. Retrieved from http://eprint.iacr.org/.

Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. 2011. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. Cryptology ePrint Archive, Report 2011/441. Retrieved from http://eprint.iacr.org/.

Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. 2012. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology (EUROCRYPT'12)*, David Pointchevaland Thomas Johansson (Eds.). Lecture Notes in Computer Science, Vol. 7237. Springer, Berlin, 446–464. DOI: https://doi.org/10.1007/978-3-642-29011-4_27

Anamaria Costache, Nigel P. Smart, and Srinivas Vivek. 2016. Faster Homomorphic Evaluation of Discrete Fourier Transforms. Cryptology ePrint Archive, Report 2016/1019. Retrieved from http://eprint.iacr.org/2016/1019.

Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. 2016. *Recovering Short Generators of Principal Ideals in Cyclotomic Rings*. Springer, Berlin, 559–585. DOI: https://doi.org/10.1007/978-3-662-49896-5_20

Ricardo Dahab, Steven Galbraith, and Eduardo Morais. 2015. Adaptive Key Recovery Attacks on NTRU-Based Somewhat Homomorphic Encryption Schemes. Cryptology ePrint Archive, Report 2015/127. Retrieved from http://eprint.iacr.org/.

Wei Dai, Yarkın Doröz, and Berk Sunar. 2014. Accelerating NTRU Based Homomorphic Encryption Using GPUs. Cryptology ePrint Archive, Report 2014/389. Retrieved from http://eprint.iacr.org/.

Valentin Dalibard. 2011. Implementing Homomorphic Encryption. St John's College. Retrieved from http://www.cl.cam.ac.uk/~ms705/projects/dissertations/2014-vd241-ihe.pdf.

I. Damgard, V. Pastro, N. P. Smart, and S. Zakarias. 2011. Multiparty Computation from Somewhat Homomorphic Encryption. Cryptology ePrint Archive, Report 2011/535. Retrieved from http://eprint.iacr.org/.

Yarkın Doröz, Yin Hu, and Berk Sunar. 2014. Homomorphic AES Evaluation using NTRU. Cryptology ePrint Archive, Report 2014/039. Retrieved from http://eprint.iacr.org/.

Yarkın Doröz, Erdinç Öztürk, and Berk Sunar. 2015. Accelerating fully homomorphic encryption in hardware. *IEEE Trans. Computers* 64, 6 (2015), 1509–1521. DOI: https://doi.org/10.1109/TC.2014.2345388

Yarkın Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. 2014. Toward practical homomorphic evaluation of block ciphers using prince. In *Financial Cryptography and Data Security (FC'14 Workshops, BITCOIN and WAHC'14), Revised Selected Papers*. 208–220. DOI: https://doi.org/10.1007/978-3-662-44774-1_17

Yarkın Doröz and Berk Sunar. 2016. Flattening NTRU for Evaluation Key Free Homomorphic Encryption. Cryptology ePrint Archive, Report 2016/315. Retrieved from http://eprint.iacr.org/2016/315.

Léo Ducas and Daniele Micciancio. 2015. *FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second*. Springer, Berlin, 617–640. DOI: https://doi.org/10.1007/978-3-662-46800-5_24

Taher El Gamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*. New York, 10–18. http://dl.acm.org/citation.cfm?id=19478.19480

Paul Erdös, Carl Pomerance, and Eric Schmutz. 1991. Carmichael's lambda function.*Acta Arith.* 4 (1991), 363–385.

Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144. Retrieved from http://eprint.iacr.org/.

Masaharu Fukase and Kenji Kashiwabara. 2015. An accelerated algorithm for solving SVP based on statistical analysis. *JIP* 23, 1 (2015), 67–80. DOI: https://doi.org/10.2197/ipsjjip.23.67

Rosario Gennaro, Craig Gentry, and Bryan Parno. 2010. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology (CRYPTO'10)*, Tal Rabin (Ed.). Lecture Notes in Computer Science, Vol. 6223. Springer, Berlin, 465–482. DOI: https://doi.org/10.1007/978-3-642-14623-7_25

Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Boneh, Dan. AAI3382729.

Craig Gentry. 2010. Computing arbitrary functions of encrypted data. *Commun. ACM* 53, 3 (March 2010), 97–105. DOI: https://doi.org/10.1145/1666420.1666444

Craig Gentry and Shai Halevi. 2011. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology (EUROCRYPT'11)*, Kenneth G. Paterson (Ed.). Lecture Notes in Computer Science, Vol. 6632. Springer, Berlin, 129–148. DOI: https://doi.org/10.1007/978-3-642-20465-4_9

Craig Gentry, Shai Halevi, and Nigel P. Smart. 2011. Fully Homomorphic Encryption with Polylog Overhead. Cryptology ePrint Archive, Report 2011/566. Retrieved from http://eprint.iacr.org/.

Craig Gentry, Shai Halevi, and Nigel P. Smart. 2015. Homomorphic Evaluation of the AES Circuit (Updated Implementation). Cryptology ePrint Archive, Report 2012/099. Retrieved from http://eprint.iacr.org/.

Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*. Springer, 75–92. DOI: https://doi.org/10.1007/978-3-642-40041-4_5

Oded Goldreich, Shafi Goldwasser, and Shai Halevi. 1997. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology (CRYPTO'97)*, Burton S., Kaliski Jr. (Ed.). Lecture Notes in Computer Science, Vol. 1294. Springer, Berlin, 112–131. DOI: https://doi.org/10.1007/BFb0052231

Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, Joe P. Buhler (Ed.). Lecture Notes in Computer Science, Vol. 1423. Springer, Berlin, 267–288. DOI: https://doi.org/10.1007/BFb0054868

Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. 2014. The past, evolving present, and future of the discrete logarithm. In *Open Problems in Mathematics and Computational Science*. Springer International Publishing, Cham, 5–36. DOI: https://doi.org/10.1007/978-3-319-10683-0_2

Antoine Joux and Jacques Stern. 1998. Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptol.* 11, 3 (1998), 161–185. DOI: https://doi.org/10.1007/s001459900042

Gabriel Kaptchuk, Matthew Green, and Aviel Rubin. 2017. Outsourcing Medical Dataset Analysis: A Possible Solution. Retrieved from http://fc17.ifca.ai/preproceedings/paper_97.pdf. (2017).

Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC.

Alhassan Khedr, P. Glenn Gulak, and Vinod Vaikuntanathan. 2014. SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. IACR Cryptology ePrint Archive 2014, 838. Retrieved from http://eprint.iacr.org/2014/838

Jinsu Kim, Moon Sung Lee, Aaram Yun, and Jung Hee Cheon. 2013. CRT-based Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2013/057. Retrieved from http://eprint.iacr.org/.

Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thom, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, DagArne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. 2010. Factorization of a 768-bit RSA modulus. In *Advances in Cryptology (CRYPTO'10)*, Tal Rabin (Ed.). Lecture Notes in Computer Science, Vol. 6223. Springer, Berlin, 333–350. DOI: https://doi.org/10.1007/978-3-642-14623-7_18

Po-Chun Kuo, Michael Schneider, Özgür Dagdelen, Jan Reichelt, Johannes Buchmann, Chen-Mou Cheng, and Bo-Yin Yang. 2011. Extreme enumeration on GPU and in clouds: How many dollars you need to break SVP challenges. In *Proceedings of the 13th International Conference on Cryptographic Hardware and Embedded Systems (CHES'11)*. Springer-Verlag, Berlin, 176–191. http://dl.acm.org/citation.cfm?id=2044928.2044944

A. K. Lenstra, H. W. Lenstra Jr., and L. Lovsz. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261, 4 (1982), 515–534. DOI: https://doi.org/10.1007/BF01457454

Tancrède Lepoint and Michael Naehrig. 2014. A Comparison of the Homomorphic Encryption Schemes FV and YASHE. Cryptology ePrint Archive, Report 2014/062. Retrieved from http://eprint.iacr.org/.

Helger Lipmaa. 2008. On the CCA1-security of elgamal and Damgård's elgamal. IACR Cryptology ePrint Archive 2008, 234. http://dblp.uni-trier.de/db/journals/iacr/iacr2008.html#Lipmaa08a

J. Loftus, A. May, N. P. Smart, and F. Vercauteren. 2010. On CCA-Secure Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2010/560. Retrieved from http://eprint.iacr.org/.

Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. 2012. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC'12)*. ACM, New York, 1219–1234. DOI: https://doi.org/10.1145/2213977.2214086

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. *On Ideal Lattices and Learning with Errors over Rings*. Springer, Berlin, 1–23. DOI: https://doi.org/10.1007/978-3-642-13190-5_1

Philip MacKenzie and Michael K. Reiter. 2001. Two-party generation of DSA signatures. In *Advances in Cryptology (CRYPTO'01)*, Joe Kilian (Ed.). Lecture Notes in Computer Science, Vol. 2139. Springer, Berlin, 137–154. DOI: https://doi.org/10.1007/3-540-44647-8_8

Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. 1996. *Handbook of Applied Cryptography*. CRC Press,Boca Raton, FL.

V. Migliore, M. Mendez Real, V. Lapotre, A. Tisserand, C. Fontaine, and G. Gogniat. 2017. Hardware/software co-design of an accelerator for FV homomorphic encryption scheme using Karatsuba algorithm. *IEEE Trans. Comput.* PP, 99 (2017), 1–1. DOI: https://doi.org/10.1109/TC.2016.2645204

C. Moore, M. O'Neill, N. Hanley, and E. O'Sullivan. 2014. Accelerating integer-based fully homomorphic encryption using Comba multiplication. In *2014 IEEE Workshop on Signal Processing Systems (SiPS'14)*. 1–6. DOI: https://doi.org/10.1109/SiPS.2014.6986063

Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop (CCSW'11)*. ACM, New York, 113–124. DOI: https://doi.org/10.1145/2046660.2046682

Erdiņç Öztürk, Yarkın Doröz, Berk Sunar, and Erkay Savaş. 2015. Accelerating Somewhat Homomorphic Evaluation Using FPGAs. Cryptology ePrint Archive, Report 2015/294. Retrieved from http://eprint.iacr.org/.

Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'99)*. Springer-Verlag, Berlin, 223–238.

Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6, Article 34 (Sept. 2009), 40 pages. DOI: https://doi.org/10.1145/1568318.1568324

R. L. Rivest, L. Adleman, and M. L. Dertouzos. 1978a. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press* (1978), 169–179.

R. L. Rivest, A. Shamir, and L. Adleman. 1978b. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (Feb. 1978), 120–126. DOI: https://doi.org/10.1145/359340.359342

C. P. Schnorr. 1988. A more efficient algorithm for lattice basis reduction. *J. Algorithms* 9, 1 (1988), 47–62. DOI: https://doi.org/10.1016/0196-6774(88)90004-1

A. Schönhage and V. Strassen. 1971. Schnelle multiplikation groβer Zahlen. *Computing* 7, 3–4 (1971), 281–292. DOI: https://doi.org/10.1007/BF02242355

P. W. Shor. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. 124–134.

N. P. Smart and F. Vercauteren. 2009. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. Cryptology ePrint Archive, Report 2009/571. Retrieved from http://eprint.iacr.org/.

Damien Stehlé and Ron Steinfeld. 2011. Making NTRU as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'11)*. Springer-Verlag, Berlin, 27–47.

Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'10)*. Springer-Verlag, Berlin, 24–43. DOI: https://doi.org/10.1007/978-3-642-13190-5_2

Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and B. Sunar. 2012. Accelerating fully homomorphic encryption using GPU. In *Proceedings of the 2012 IEEE Conference on High Performance Extreme Computing (HPEC'12)*. 1–5. DOI: https://doi.org/10.1109/HPEC.2012.6408660

Wei Wang and Xinming Huang. 2013. FPGA implementation of a large-number multiplier for fully homomorphic encryption. *IEEE International Symposium on Circuits and Systems*. IEEE, 2589–2592.

David Wu and Jacob Haven. 2012. *Using Homomorphic Encryption for Large Scale Statistical Analysis*. Technical Report. Leland Stanford Junior University.

Received January 2016; revised July 2017; accepted July 2017