**0368.4162 Foundations of Cryptography** January 9, 2020

# Lecture 10: Fully-Homomorphic Encryption

*Lecturer: Nir Bitansky*

## 1 Previously on Foundations of Crypto

In the previous two lectures, we learned about multi-party computation. We've seen that already in the 80's cryptographers have managed to prove a very strong completeness theorem — *assuming oblivious transfer any function can be securely computed*. That may seem as a good place to stop the course, what else could we want? As we've already mentioned in previous lectures this is hardly the case because the latter completeness theorems are not sensitive to resources — they require lot's of interaction and computation, which are sometimes not available.

In the next lectures, we'll encounter several such scenarios and introduce new type of tools that would solve them. The problems that we'll look at are mostly motivated by today's reality where most of our data (including private data) is stored and processed on various clouds.

## 2 Fully-Homomorphic Encryption

Say you want to store your private data $x$ on the web, so that on one hand, the server doesn't learn the contents of your data, but on the other, you can still ask the server to perform any computation $f$ on the data, so that you would eventually get $f(x)$. Naturally, you want the communication and computation you have to invest to be proportional to the input-output size, and the description of $f$, but not the time it takes to perform the computation.

Fully-homomorphic encryption (FHE) [RSA78] achieves exactly these objectives. It basically provides a way *to compute under the encryption*.

**Definition 2.1** ($F$-Homomorphic Encryption)**.** *Let $F = \{F_n\}$ be a collection of $n^{O(1)}$-size circuits with Boolean output. A secret-key, CPA-secure, bit encryption scheme $(E, D)$ is $F$-Homomorphic if there exists an algorithm Eval satisfying:*

- **Homomorphism:** *for any $sk \in \{0,1\}^n$, $f \in F_n$,*

$$D_{sk}(Eval_f(E_{sk}(x_1), \ldots, E_{sk}(x_\ell))) = f(x_1, \ldots, x_\ell) \ .$$

- **Compactness:** *the output of $Eval_f$ is of length $n$ (doesn't grow with the complexity of $f$).*

*The scheme is* **fully homomorphic** *if it is homomorphic for any collection $F$ of polynomial-size circuits.*

**Remark:** The restriction to the secret key setting is actually w.l.o.g. for fully-homomorphic encryption. Indeed, there's a generic transformation from secret to public key [Rot11].

Schemes that are homomorphic for a specific operation, e.g. addition, have been known for quite some time. In fact, most public-key schemes, such as RSA or Diffie-Hellman/Elgamal, are algebraic enough to admit some homomorphism. Constructing fully-homomorphic encryption was an open problem for three decades, and was in fact believed by many to be implausible. This changed in 2009 with a breakthrough result by Craig Gentry [Gen09] who provided the first candidate for homomorphic encryption. His initial construction already contained ingenious ideas, but was based on quite non-standard hardness assumptions. In the following years, fully-homomorphic encryption was based on standard assumptions [BV11], made much more efficient, and considerably simplified. Today, we will see one such construction by Gentry, Sahai, and Waters [GSW13], with the main objective being simplicity (and not necessarily efficiency or security).

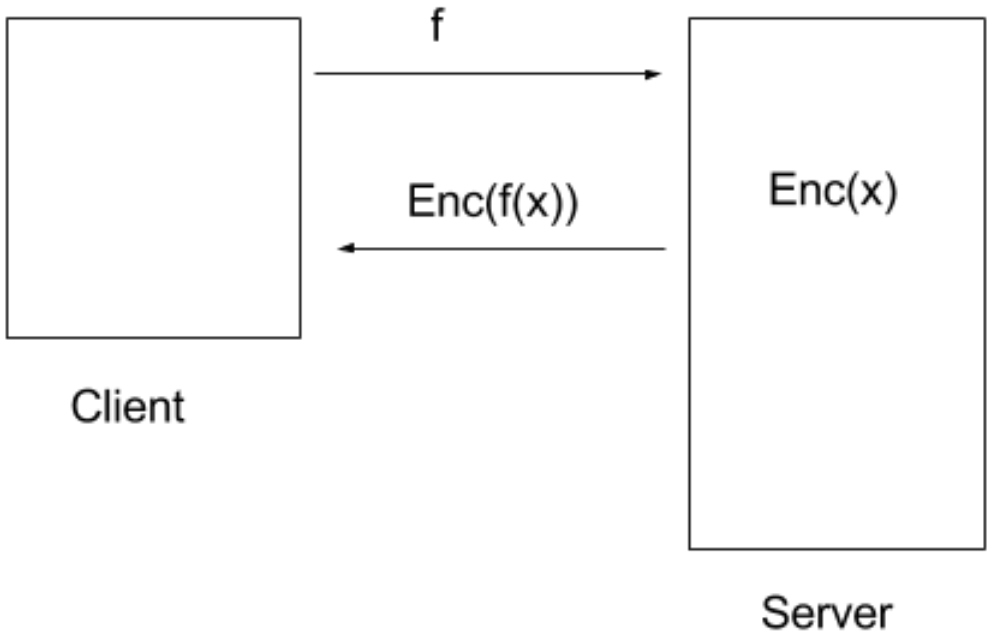


**Figure 1**: FHE allow to delegate computations to be performed over encrypted input. The total communication and computation by the client depend on $|f| + |f(x)|$ but not on $time(f, x)$

**Gentry's Blueprint.** We will follow the blueprint suggested in Gentry's construction, and utilized also in its followups. This includes two main steps:

1. Construct a scheme for shallow circuits (where we will address how shallow later on).
2. Show how to convert any such scheme to a scheme for general circuits.

# 3 Homomorphic Encryption for Shallow Circuits from LWE

The construction we'll see is based on the hardness of a specific algebraic assumption called *Learning with Errors* (LWE), which is a cousin of the Learning Parity with Noise Problem (LPN) that we've already met.

**The Learning with Errors Assumption [Reg05]:** Let $q = q(n)$ be a prime, and let $B = B(n) \ll q(n)$. The (decisional) $LWE_{q,B}$ assumption says that there is an efficiently samplable distribution $\chi = \chi_n$ on integers in $[-B, B]$ such that the following two oracles are computationally indistinguishable:

1. The oracle $NLE_{\chi,s}$ that is parameterized by a random $s \leftarrow \mathbb{Z}_q^n$, and returns noisy linear equations in $s$ of the form:
$$a, \langle a, s \rangle + e \text{ where } a \leftarrow \mathbb{Z}_q^n \text{ and } e \leftarrow \chi \ .$$
2. The random oracle $R$ that returns random samples
$$a, b \text{ where } a \leftarrow \mathbb{Z}_q^n, b \leftarrow \mathbb{Z}_q \ .$$

Like the LPN problem, this problem is related to the problem of decoding random linear codes (a collection of $m \gg n$ samples can be viewed as a noisy version of the code word $As$, for a random linear code whose generator matrix is $A$). Intuitively, the problem becomes easier as the ratio $B/q$ decreases (in the extreme cases, when the ratio is zero, the problem is solved by gaussian elimination, and when the ratio is $1/2$, the problem is information theoretically impossible to solve).

This problem, however, has so far turned out to be much more versatile than LPN. Indeed, the transition to a large moduli $q$ and different error distributions allows for geometric interpretations of the problem and leads to strong connections to problems on lattices. Remarkably there are also reductions between the (average-case) hardness of the problem and the worst-case hardness of classical approximation problems on lattices such as (the gap version of) the shortest vector problem (where as expected the approximation factors are related to the noise-to-moduli ratio).

**A Useful Abstraction.** To construct homomorphic encryption based on LWE, it will be convenient to think about the following abstraction.

**Definition 3.1** (Pseudorandom Approximate Hyperplane (PAH))**.** *A $(q, B)$-PAH is given by PPT algorithms $(G, H)$ where $G(1^n)$ generates a secret-key vector $s \in \mathbb{Z}_q^n$ and $H_s$, given additional randomness, samples vectors $h \in \mathbb{Z}_q^n$ so that*

- *for any $h$ sampled from $H_s$, $\langle h, s\rangle \in [-B, B]$.*
- *The oracle $H_s$ is computationally indistinguishable from an oracle that samples random vectors in $\mathbb{Z}_q^n$.*
- $s_1 = \lceil q/2 \rceil$.[1]

**Claim 3.2.** *$LWE_{q,B}$ implies a $(q, B)$-PAH.*

*Proof Sketch.* $G(1^n)$ outputs $s = (\lceil q/2 \rceil, s')$ for an LWE secret $s' \in \mathbb{Z}_q^{n-1}$. $H_s$ outputs $(\frac{as'+e}{\lceil q/2 \rceil}, -a)$. □

We will prove the following:

**Theorem 3.3** ([GSW13])**.** *Let $q \approx 2^{\sqrt{n}}$ and $B \approx \sqrt{q}$. Then a $(q, B) - PAH$ implies a homomorphic scheme for circuits of depth $n^{.49}$.*

**Warmup: Homomorphism for Additions (mod 2).** As a warmup, let us use our PAH to construct a scheme that supports an arbitrary polynomial number of additions mod 2:

- The secret key will be a secret PAH vector $s \in \mathbb{Z}_q^n$.
- To encrypt $b \in \{0, 1\}$ sample $h \leftarrow H_s$ and output $c = h + (b\rceil, 0, \ldots, 0)$.
- To decrypt $c$, output 0 if $\langle c, s\rangle \in [-q/4, q/4]$ or 1 otherwise.
- To homomorphically add $c$ and $c'$ output $c + c'$.

CPA security of this scheme easily follows by the fact that it has pseudorandom ciphertexts.

To see correctness, it is useful to first imagine that $H_s$ generates vectors $h$ that are truly orthogonal to $s$. In this case,

$$\langle c, s\rangle = \langle h, s\rangle + bs_1 = 0 + b\lceil q/2 \rceil \mod q \ ,$$

which is in $[-q/4, q/4]$ iff $b = 0$.

Furthermore, for any $c_1, \ldots, c_\ell$ encrypting $b_1, \ldots, b_\ell$,

$$\langle \sum c_i, s\rangle = \langle \sum h_i, s\rangle + s_1 \sum b_i = 0 + \lceil q/2 \rceil \sum b_i \mod q \approx_\ell \lceil q/2 \rceil \bigoplus_i b_i \ ,$$

[1] This requirement is not essential ($s \neq 0$ suffices), but it will simplify a bit our constructions.

where $\approx_\ell$ means up to an additive error of $\ell = n^{O(1)} \ll q/4$, meaning that this will decrypt to $\bigoplus_i b_i$.

Going back to reality, we cannot really choose the vectors $h$ to be orthogonal, as then they won't be pseudorandom (they all come from an $(n-1)$-dimensional space, and this will be detected after $\approx n$ samples). However, the same idea works when $h$ is only close to being orthogonal — the above equations simply become approximate. In particular, $\langle h_i, s\rangle \approx_B 0$ and $\langle \sum h_i, s\rangle \approx_{\ell B} 0$. Since $\ell B \ll q/4$ decryption remains correct.

**Supporting Multiplication via The Eigen-Vector Method.** To support general circuits, we need to support a universal set of operations. For this, we will want a scheme that supports both $+, \times \mod 2$. Thinking about our warmup additive scheme, there's no obvious way to homomorphically multiply two ciphertexts.[2]

The high-level idea would be to generalize the scheme so that any ciphertext encrypting $b$ is a matrix $C$ instead of vector $c$. We'll replace the invariant $c^T s \approx bs_1$ with $Cs \approx bs$. Namely, the secret key $s$ will be an approximate eigen vector of $C$ with the encrypted message $b$ as an eigen value.

As before let's imagine for a second that the secret key is truly an eigen vector (rather than approximate). Then, we have

$$(C + C')s = (b + b')s \qquad \text{and} \qquad (CC')s = bb's \ .$$

meaning we can make any (polynomial) number of these operations.

Again, we can not hope for security with exact eigen vectors, as these are easy to find using gaussian elimination. We will want to show that this idea can still be applied with approximate eigen-vectors — we would like to construct our matrices $C$ (with approximate eigen-vector $s$) to be pseudorandom.

**1st Attempt.** It seems we've already developed the needed tools to achieve the above goal. We will construct the scheme as follows:

- The secret key will be a vector $s \in \mathbb{Z}_q^n$ for a PAH.
- An encryption of $b$ will be generated by sampling $H = (h_1, \ldots, h_n)^T$ where $h_i \leftarrow H_s$ and outputting $C = H + bI$.
- To decrypt $C$, compute $v = Cs$. Output 0 if $v_1 \in [-q/4, q/4]$ or 1 otherwise.
- To homomorphically add $C$ and $C'$ output $C + C'$.
- To homomorphically multiply $C$ and $C'$ output $CC'$.

CPA security of this scheme follows as before by the fact that $H$ and thus also $C$ are pseudorandom.

We now analyze correctness. We define the *error* $e(C)$ in a ciphertext $C$ encrypting $b$ with $s$ and its magnitude $\|e(C)\|$ as

$$e(C) := Cs - bs \qquad \|e(C)\| := \|e(C)\|_\infty := \max_i |e(C)_i| = \min\{\beta \mid e(C) \in [-\beta, \beta]^n\} \ .^3$$

For an arithmetic function $f$, we will want to understand the error in a ciphertext $C$ encrypting $f(x)$ that is the result of homomorphically evaluating $f$ on fresh encryptions of $x_1, \ldots, x_\ell$, aiming that $\|e(C)\| \ll q/4$. Indeed, if that's, then for $v = Cs = e(C) + f(x)s$ it holds that $v_1 \approx_{q/4} f(x)\lceil q/2 \rceil$, and decryption will be correct.

We examine the error:

- In any fresh (i.e., generated by the encryption algorithm) ciphertext $C$:

$$\|e(C)\| = \|Hs\| \le B \ .$$

[2] Two natural rather natural ways of multiplying vectors is via inner product or outer product. The first loses the geometric relation with the secret key $s$. The second actually makes sense, but results in a ciphertext vector that is quadratically larger. There are ways to *relinearize* it [BV11], but today we'll take a different approach.

[3] Formally, $e(C) = e_{s,b}(C)$ is defined with respect to $s, b$.

- Given two (possibly not fresh) ciphertexts $C$ and $C'$ encrypting $b$ and $b'$:

$$\|e(C+C')\| = \|(C+C')s - (b+b')s\| = \|e(C)+e(C')\| \leq \|e(C)\| + \|e(C')\| \ .$$

$$\|e(CC')\| = \|CC's - bb's\| = \|C(b's+e(C')) - bb's\| = \|b'e(C) + Ce(C')\| \ .$$

As we can see, while adding two ciphertexts with small error results in a ciphertext with small error, this is not the case for multiplication. Indeed, if the matrix $C$ is pseudorandom, then it is likely to be the case that $\|Ce(C')\| \notin [-q/4, q/4]$.

**Solution: Binary Encoding.** There is a surprisingly naïve solution to the last problem. Specifically, it would have been great if we could only encode the matrix $C$, containing large integers in $\mathbb{Z}_q$, as a new matrix that only has small entries. Hmmm... do we have such an encoding? We do, binary representation!

For every $c \in \mathbb{Z}_q$, let $bin(c) \in \{0,1\}^{1\times \log q}$ be its binary representation.[4] Consistently, for a matrix $C \in \mathbb{Z}_q^{m\times n}$, with entries $c_{ij}$, let $bin(C) \in \{0,1\}^{m\times n\log q}$ be the binary matrix representing each entry $bin(c_{ij})$.

Roughly speaking, instead of multiplying $C$ by $C'$, we would like to multiply $bin(C)$ with $C'$. However, we first need to slightly augment the scheme as $bin(C)$ may lose the geometric relation between $C$ and $s$, in fact, the dimensions don't even work out. We will consider the inverse operation to $bin$, which takes the binary representation and turns it into back into representation in $\mathbb{Z}_q$:

$$Q = \begin{pmatrix} 1 & & & & & & & \\ 2 & & & & & & & \\ \vdots & & & & & & & \\ 2^{\log q} & & & & & & & \\ & 1 & & & & & & \\ & 2 & & & & & & \\ & \vdots & & & & & & \\ & 2^{\log q} & & & & & & \\ & & \ddots & & & & & \\ & & & \ddots & & & & \\ & & & \vdots & & & & \\ & & & & \ddots & & & \\ & & & & & 1 & & \\ & & & & & 2 & & \\ & & & & & \vdots & & \\ & & & & & 2^{\log q} & & \end{pmatrix}_{n\log q \times n}$$

Note that for any $C \in \mathbb{Z}_q^{n\log q\times n}$,

$$bin(C)Q = C \ .$$

We will now tweak our encryption as follows:

- The secret key will still be a vector $s \in \mathbb{Z}_q^n$ for a PAH.
- An encryption of $b$ will be generated by sampling $H = (h_1, \ldots, h_{n\log q})^T$ where $h_i \leftarrow H_s$ and outputting $C = H + bQ$.

[4] Formally, we should consider $\lceil \log q \rceil$.

- To decrypt $C$, compute $v = Cs$. Output 0 if $v_1 \in [-q/4, q/4]$ or 1 otherwise.
- To homomorphically add $C$ and $C'$ output $C + C'$.
- To homomorphically multiply $C$ and $C'$ output $bin(C)C'$.

**Analysis.** The security of the augmented schemes follows just as the security of the previous scheme. We now analyze correctness. We redefine the *error* $e(C)$ in a ciphertext $C$ encrypting $b$ as

$$e(C) := Cs - bQs \quad .$$

We reexamine the error:

- As before, in any fresh ciphertext $C$:

$$\|e(C)\| = \|Hs\| \leq B \ .$$

- Given two (possibly not fresh) ciphertexts $C$ and $C'$ encrypting $b$ and $b'$:

$$\|e(C + C')\| = \|(C + C')s - (b + b')Qs\| = \|e(C) + e(C')\| \leq \|e(C)\| + \|e(C')\| \ .$$

$$\begin{aligned}\|e(bin(C)C')\| =& \|bin(C)C's - bb'Qs\| = \|bin(C)(b'Qs + e(C')) - bb'Qs\| = \\ & \|b'Cs + bin(C)e(C') - bb'Qs\| = \|b'e(C) + bin(C)e(C')\| \leq \|e(C)\| + n\log q\|e(C')\| \ .\end{aligned}$$

Eventually, we would like to homomorphically evaluate an arbitrary arithmetic function $f$. We can think w.l.o.g on the circuit describing the function as consisting of interchanging all-addition layers and all-multiplication layer, the error in the resulting ciphertext $C$ will depend on the number of multiplication layers, which is bounded by the depth $d$. Specifically it is dominated by

$$\|e(C)\| \leq B\,(n\log q)^{O(d)} \ ,$$

and thus for $d \leq n^{0.49}$,

$$\|e(C)\| \leq o(2^{\sqrt{n}}) \ll q/4 \ ,$$

implying that we will obtain the correct result when decrypting.

# 4 Bootstrapping: From Shallow Circuits to Arbitrary Circuits

The latter scheme actually allows to construct what is known as a leveled homomorphic encryption — if we know a bound on the depth $d$ of the circuits to be evaluated ahead of time, we can choose our parameters $n, q$ such that $d \leq n^{0.49}$. Going back to Gentry's blueprint, we would now like to show how to turn the scheme we got into a fully homomorphic encryption, where an apriori bound on the depth is not known.

Here Gentry had a simple, but ingenious observation: *to correctly evaluate arbitrary functions $f$, it is enough to be able to correctly evaluate* the scheme's own decryption circuit plus one universal operation.

Formalizing this requires an extra *circular security* assumption:

**Theorem 4.1** ([Gen09])**.** *Consider the function*

$$f^D_{C,C'}(s) = D_s(C) \quad NAND \quad D_s(C') \ .$$

*Then any homomorphic scheme $(E, D)$ for $f$ that is secure in the presence of an encryption $E_s(s)$ can be made fully homomorphic.*

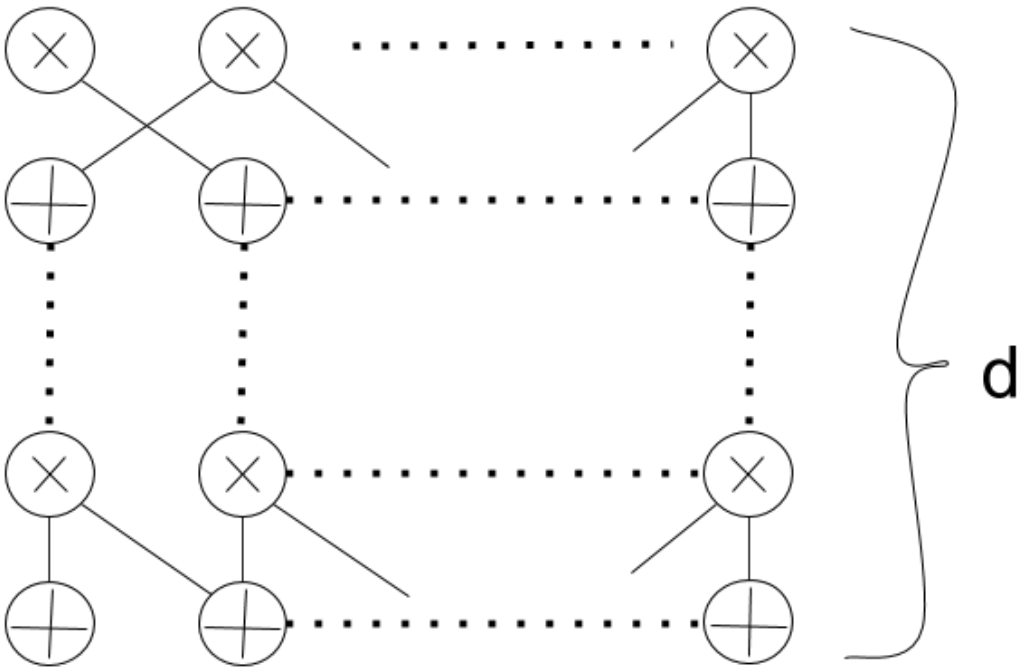


**Figure 2**: A layered arithmatic circuit of depth $d$

The idea behind this theorem is to publish $E_s(s)$ as an evaluation key. Then every time we evaluate $f^D_{C,C'}$ homomorphically on $E_s(s)$, with "decryptable" $C, C'$, we get a decryptable encryption of $D_s(C) \quad NAND \quad D_s(C')$ and can move on this way through the entire circuit.

*Did our scheme support its own decryption circuit?*

Recall, that our scheme supported circuits of depth $n^{0.49}$, and our decryption process was:

- To decrypt $C$, compute $v = Cs$. Output 0 if $v_1 \in [-q/4, q/4]$ or 1 otherwise.

This could be simplified to checking if $\langle c_1, s\rangle \in [-q/4, q/4]$. Computing an inner product of vectors of dimension $n$ over $\mathbb{Z}_q$ can be done in depth $\log n$ if we allow arithmetic operations $\mod q$. However, allowing only arithmetics over $\mathbb{Z}_2$, leads to an overhead of polylog$(q)$, and will thus exceed our allowed bound $d \ll \log q$.

What saves us is the fact that we only care about the two most significant bits of $\langle c_1, s\rangle$. Specifically, we're given encryptions of the bits of $bin(s)$, and we'd like to compute the first two bits of

$$\langle bin(s)Q, c_1\rangle = \langle bin(s), Qc_1\rangle = \sum_i bin(s)_i\gamma_i \mod q,$$

where $\gamma := Qc_1 \mod q$. Overall, we're adding at most $m = n\log q$ numbers $\mod q$. For every $i$, we can consider $\tilde{\gamma}_i$, where all but the $2\log m$ most significant bits of $\gamma_i$ are zeroed out, as well as $\tilde{q}$, where all but the $2\log m$ most significant bits of $q$, are zeroed out. We will compute

$$\sum bin(s)_i\tilde{\gamma}_i \mod \tilde{q} \ ,$$

which can be done in depth polylog$(m)$ = polylog$(m)$.

Note that

$$\left|\sum bin(s)_i\gamma_i - \sum bin(s)_i\tilde{\gamma}_i\right| \leq m \max_i |\gamma_i - \tilde{\gamma}_i| \leq m\frac{q}{m^2} = \frac{q}{m} \ .$$

It is left to bound the effect of taking modulo $\tilde{q}$ instead of $q$. Note that $\Gamma := \sum bin(s)_i\tilde{\gamma}_i \leq m\tilde{q}$. We thus have:

$$\Gamma = t\tilde{q} + r = tq + t(\tilde{q} - q) + r \ ,$$

where $t \leq m$ and $r < \tilde{q}$. We thus have:

$$|\Gamma \mod \tilde{q} - \Gamma \mod q| = |t(\tilde{q} - q)| \leq m\frac{q}{m^2} \ .$$

The security of the scheme clearly holds so long that the original scheme is secure given $E_s(s)$. Getting FHE from standard assumptions and without making a circular-security assumption is still an open problem.

# References


[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Yuval Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2011.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.