CSC 2419: Lattice-based Cryptography **Fall 2025**

Lecture 7: Secure Computation: FHE and LFE

Instructor: Akshayaram Srinivasan *Scribe: Mani Mehdipoor*

Date: 2025-11-03

7.1 Recap

In the previous lecture, we discussed Non-Interactive Zero-Knowledge (NIZK) proofs, in which a prover convinces a verifier that a statement is true without revealing any information about the witness and without having multiple back-and-forth communications with the verifier.

Specifically, we saw how to construct NIZK from 3-message Zero-Knowledge and Correlation-Intractable (CI) hashing. Recall that a 3-message Zero-Knowledge proof follows the "Commit–Challenge–Response" structure. To ensure soundness in the non-interactive setting, we needed a CI hash function to avoid generating the bad challenge strings $b_1^\star, \ldots, b_n^\star$, for which it would be possible to cheat the verifier on every challenge. We discussed how to construct such a CI hash function using Rate-1 FHE.

7.2 Definition of Secure Computation

As usual, we consider two polynomially-bounded parties: Alice and Bob. Alice holds a private input $x \in \mathcal{X}$, and Bob holds a private input $y \in \mathcal{Y}$. These are common inputs to some function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ which they wish to compute through an interactive protocol, by exchanging rounds of messages. In this lecture, we only consider protocols that involve two messages: Alice sends the first message msg_1 to Bob, and then Bob responds with a message msg_2 to Alice.

Correctness. At the end of the protocol, Alice outputs a value z, which should equal $f(x, y)$ except with negligible probability.

Security. Informally, we want Bob to learn no information about x, and Alice to learn no information about y except what can be revealed from x and $f(x, y)$. Note that we could have a function such as $f(x, y) = x \oplus y$, in which case x and $f(x, y)$ would reveal y completely; this would still be considered acceptable. We do not care what the safe set of functions to compute is; in our definition, we only care that Alice learns nothing beyond what we can learn from just x and $f(x, y)$.

Formally, for Alice's security, we require that, for all $x, x' \in \mathcal{X}$,

$$\{\mathrm{msg}_1 \leftarrow \mathrm{Alice}(x)\} \approx_c \{\mathrm{msg}_1 \leftarrow \mathrm{Alice}(x')\}$$

To formalize Bob's security, let us first consider two naive attempts.
Attempt 1. One might require that for all $y, y' \in \mathcal{Y}$, $\{\mathrm{msg}_2 \leftarrow \mathrm{Bob}(\mathrm{msg}_1, y)\} \approx_c \{\mathrm{msg}_2 \leftarrow \mathrm{Bob}(\mathrm{msg}_1, y')\}$. However, this has a trivial distinguishing attack: If $f(x, y) \neq f(x, y')$, then Alice can distinguish the two simply by finding the outputs.

Attempt 2. We might instead restrict to y, y' such that $f(x, y) = f(x, y')$. While this avoids the previous problem, it is not an ideal definition since it may not be possible to find such parts y, y' efficiently (for example, when f is a one-way function).

To obtain the correct formal definition, we are going to use the simulation paradigm, which we considered last lecture for ZK proofs. For secret input $x \in \mathcal{X}$, we define the view of Alice as $\text{View}_A = (x, r, \text{msg}_2)$, where r is the internal randomness of Alice. We say that Bob's security holds if there exists a PPT simulator Sim such that for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$,

$$\{\text{View}_A(x, r, \text{msg}_2)\} \approx_c \{\text{Sim}(x, f(x, y))\}$$

Since Alice's view can be generated (up to computational indistinguishability) by a simulator which is polynomial-time and only gets input x and $f(x, y)$, this shows that Alice's real word view does not reveal any information about y except what can already be learnt from x and $f(x, y)$.

Here, we assumed that both Alice and Bob follow the exact protocol instructions, but may try to learn extra information from what they can see. This is known as the semi-honest model.
In the malicious model, we no longer hold this assumption, meaning that Alice or Bob could deviate from the protocol by sending fake messages that are not consistent with their inputs or the protocol's rules. Zero-knowledge proofs actually give us a way to upgrade from the semi-honest model[1] to a malicious model. We could ask each party to also generate a ZK proof showing that they correctly computed their messages according to the protocol and their private inputs. The soundness property of the ZK proof ensures that the messages truly correspond to a valid computation; so, if a party tries to cheat, they would not be able to generate a valid proof. At the same time, the zero-knowledge property ensures that no information is revealed about the private inputs of the parties, preserving security. In this way, ZK effectively binds the parties to follow the protocol without revealing any secrets or relying on trust. This allows a protocol that was only secure against semi-honest adversaries to be easily upgraded to security against malicious adversaries, which actually illustrates one of the key applications of ZK proofs in cryptography.

Goal. The goal of this lecture is to use lattice-based tools to construct a protocol for this task with very low communication. By low communication, we mean that we want the total size of messages exchanged, i.e. $|\text{msg}_1|+|\text{msg}_2|$, to be as small as possible.

7.3 Alice's input is short

We start with a scenario where Alice holds a secret input $x \in \{0,1\}^n$ and Bob holds a secret input $y \in \{0,1\}^N$, with $N \gg n$. In other words, Bob's input here is much larger than Alice's.

Let us first consider the most trivial and *insecure* protocol: Alice sends x to Bob, Bob computes $f(x, y)$ using his input y, and sends the result back to Alice. In this case, the communication cost is $|\text{msg}_1|+|\text{msg}_2|= |x|+|f(x, y)|= n + 1$, which is actually optimal (even in the communication-complexity sense) for some functions, such as database lookups.

Now, let us think about how to make this insecure protocol secure. Fully Homomorphic Encryption (FHE) provides a natural solution. Consider the function $f[y] : \mathcal{X} \to \{0, 1\}$ which has y hardcoded and takes in $x \in \mathcal{X}$ as input and outputs $f(x, y)$. We first try the following protocol:

[1]Technically speaking, we need a slightly stronger notion than semi-honest for this upgrade to work. In the stronger notion, we require security against adversaries who follow the protocol but might choose their random coins arbitrarily. This notion is called semi-malicious security.

1. Alice sends an FHE encryption of her input $ct = \text{FHE-ENC}(x)$ to Bob.
2. Using the publicly computable algorithm FHE-EVAL, Bob computes $f[y]$ directly on the ciphertext ct of x.
3. Bob sends the resulting ciphertext, which encrypts $f[y](x)$, back to Alice.
4. Alice decrypts using her secret key to recover $f(x, y)$.

Correctness. The correctness of the protocol follows immediately from the correctness of the underlying FHE scheme.

Security. Security of Alice is easy. The semantic security of FHE says that: for all $x, x' \in \mathcal{X}$,

$$\{\text{FHE-ENC}(x) \leftarrow \text{Alice}(x)\} \approx_c \{\text{FHE-ENC}(x') \leftarrow \text{Alice}(x')\},$$

which is exactly what we want for the security of Alice.

The security of Bob is more subtle. It would amount to showing that FHE-EVAL$(f[y], ct)$ is actually computationally indistinguishable to a fresh encryption of $f(x, y)$. If this holds, then we can ensure security for Bob because we can consider a simulator that encrypts $f(x, y)$ as msg_2 and samples randomness r to produce a view that is computationally indistinguishable to Alice's real-world view.

However, in the case of the GSW-based FHE scheme, this equivalence does not exactly hold. Let us first recall the GSW FHE scheme:
The public key is $B = \begin{bmatrix} A \\ s^T A + e^T \end{bmatrix}$, and encryption is computed by sampling a binary matrix R and outputting $ct = BR + x \otimes G$, where G is the gadget matrix. Here $x \in \{0, 1\}^n$, so we consider the tensor product $x \otimes G$ for shorthand.

During the FHE Evaluation, the output ciphertext has the form $BR' + f(x, y) \cdot G$, for some noise R' determined by the circuit of $f[y]$. If we do an addition gate, R' is just the addition of R_1 and R_2. However, for multiplication gates, things become more complicated, and we actually encounter a problem because R' will be something like $R_1 G^{-1}(c_2) + y \cdot R_2$, which actually leaks some information about y. Specifically, since Alice knows both s^T and e^T, she can potentially try to extract some information about y from the structure of R'. Therefore, for the standard GSW encryption that we considered, the evaluated ciphertext FHE-EVAL$(f[y], ct)$ is not actually indistinguishable from FHE-ENC(f(x,y)). So, we must modify our approach. Specifically, we need to somehow sanitize the noise of this ciphertext so that it does not contain any information about y. This technique is known as noise flooding.

7.3.1 Noise Flooding

Note that (bootstrapping as needed) R' is actually a matrix that has a bounded L-infinity norm; let's say that $\|R'\|_\infty \leq B$, which we denote as $R' \in [-B, B]$. Let λ denote the security parameter. To flood the noise, we now introduce a larger bound B' chosen as $B' = B \cdot \lambda^{\omega(1)}$, which is super-polynomially larger than B. Here, we are assuming that B' is still below the noise bound.

Within the FHE-EVAL procedure, Bob then samples a fresh random matrix $R'' \in [-B', B']$ (i.e. a uniform matrix with entries in $[-B', B']$) and adds BR'' to the ciphertext to obtain a new ciphertext

$$B(R' + R'') + f(x, y) \cdot G$$

Intuitively, by doing this, we are "flooding" the bounded noise R' with a much larger random noise R'', making the ciphertext statistically independent of y, which explains the name of the technique.

Security. To show security, our claim is that $R' + R''$ is statistically close to uniform distribution over $[-B', B']$. This would show that the new noise is statistically independent of y, which would ensure the security of Bob.

The argument closely parallels the one we used for spooky rounding. As long as R'' does not fall within the small bad intervals $[-B', -B' + B]$ or $[B' - B, B']$ (which could cause $\|R' + R''\|_\infty$ to exceed B'), the sum $R' + R''$ will be uniform over $[-B', B']$. Thus, the statistical distance between the true distribution of $R' + R''$ and uniform is $2B/2B' = B/B'$, which is negligible since B' was chosen to be superpolynomially larger than B.

Communication Cost. We can essentially achieve a communication cost close to $n + 1$ by encrypting a short PRG seed and XORing the PRG output with x, which makes Alice's communication cost of order $n + \text{poly}(\lambda)$, and with ciphertext compression, this can be reduced to $1 + \text{poly}(\lambda)$.

Until now, this is the only known approach to achieve secure computation with communication costs close to the size of Alice's input. Other known approaches have communication costs that grow with the circuit size or Bob's input.

7.4 Bob's input is short

Let us now consider the opposite setting. Suppose Alice holds a secret input $x \in \{0,1\}^N$ and Bob holds a secret input $y \in \{0,1\}^n$, where $N \gg n$. In this case, the trivial and insecure protocol would just have Bob send y to Alice, and then have Alice compute $f(x, y)$ using her input x. As in the last scenario, our goal is to construct a secure protocol that achieves a communication cost close to that of this trivial, insecure approach. This problem is actually known as *Laconic Function Evaluation (LFE)*, introduced by Quach, Wee, and Wichs in [QWW18].

7.4.1 Key-Equation

A central tool for constructing LFE protocols from lattices is the *Key Equation* which we now consider.

Consider any input $y = y_n \cdots y_2 y_1 \in \{0,1\}^n$ and a function $f : \{0,1\}^n \to \{0,1\}$. Further, suppose that we have a matrix

$$C = [A_1 + y_1 G \parallel A_2 + y_2 G \parallel \cdots \parallel A_n + y_n G].$$

where $A_1, \ldots, A_n$ are public matrices and G is the gadget matrix.
Then, we want to show that there exists a "low ℓ_∞-norm" matrix $H_{f,y}$ that depends both on f and y such that

$$C \cdot H_{f,y} = A_f + f(y) \cdot G,$$

where the matrix A_f depends only on f and the public matrices $A_1, \ldots, A_n$, but not the input y.
Here, by "low ℓ_∞-norm" we mean low norm in the sense that the noise grows similarly to the noise growth of the randomness R in the GSW encryption.

Proof: Let us prove the Key-Equation using induction on the depth of the function f.

The base case is the input level. Meaning that $f(y) = y_i$ for some i; without loss of generality, suppose that $f(y) = y_1$. For this case, the relation holds trivially. We can take $H_{f,y} = \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix}$, and $A_f = A_1$. Then $H_{f,y}$ has low norm by construction, and we obtain $C \cdot H_{f,y} = A_1 + y_1 G = A_f + f(y)G$, as desired.

Assume the claim holds for all functions of depth at most $d-1$, and consider a gate at level d. Let the gate take as input two wires $g_1(y)$ and $g_2(y)$ computed at level $d-1$. Suppose that $H_{g_1,y}$ and $H_{g_2,y}$ are the low-norm matrices generating $A_{g_1} + g_1(y)G$ and $A_{g_2} + g_2(y)G$, respectively, each with norm bounded by B.

First, let's consider an addition gate; i.e. the gate computes $f(y) = g_1(y) + g_2(y)$. Then, we can take $H_{f,y} = [H_{g_1,y} \| H_{g_2,y}] \cdot \begin{bmatrix} I \\ I \end{bmatrix}$ and $A_f = A_{g_1} + A_{g_2}$. Here, $H_{f,y}$ has norm bounded by $2B$, and

$$C \cdot H_{f,y} = A_{g_1} + A_{g_2} + (g_1(y) + g_2(y))G = A_f + f(y)G,$$

as we require.

The interesting case is multiplication; i.e., the gate computes $f(y) = g_1(y) \cdot g_2(y)$. This behaves similarly to multiplication in the GSW scheme. We take $A_f = -A_{g_1} G^{-1}(A_{g_2})$ and $H_{f,y} = [H_{g_1,y} \| H_{g_2,y}] \begin{bmatrix} -G^{-1}(A_{g_2}) \\ g_1(y) I \end{bmatrix}$. Then A_f depends only on f and not on y, while $H_{f,y}$ has norm bounded by $B \cdot \mathrm{poly}(\lambda)$ (i.e., it grows comparably to R in GSW). Further, we have

$$\begin{aligned} C \cdot H_{f,y} &= -(A_{g_1} + g_1(y)G) \cdot G^{-1}(A_{g_2}) + g_1(y)(A_{g_2} + g_2(y)G) \\ &= -A_{g_1} G^{-1}(A_{g_2}) - g_1(y) A_{g_2} + g_1(y) A_{g_2} + g_1(y) g_2(y) G = A_f + f(y)G, \end{aligned}$$

which is exactly what we required. This proves the equation for a multiplication gate.

By induction, the claim holds for all functions f, completing the proof of the Key Equation. ■

Despite its simplicity, the Key-Equation (introduced in [BGG+14]) has become one of the most versatile and widely used tools in lattice-based cryptography over the past decade, with countless applications. We will also revisit it in later lectures.

7.4.2 LFE from Key-Equation

We will now use the Key-Equation above to get LFE. We will do this in several steps, starting from a simple but insecure construction and gradually adding security at each step.

Setup. Let the common reference string (CRS) consist of the public matrices $A_1, \ldots, A_n$ appearing in the Key Equation. Recall that Alice holds a large secret input $x \in \{0,1\}^N$ and Bob holds a smaller secret input $y \in \{0,1\}^n$, with $N \gg n$. The goal is to allow Alice to compute $f(x,y)$ securely, with low communication. Since Alice has to send the first message msg_1, an FHE solution, like before, would not work here. So, we take a different approach.

Step 1. This step gives an insecure protocol, which we will later patch.
Alice first sets the function $f[x] : \mathcal{Y} \to \{0,1\}$ to be the function that takes $y \in \mathcal{Y}$ as input and outputs $f(x,y)$. She then uses Key-Equation to compress this function into A_f and sends A_f to Bob. Note that since A_f depends only on f and the CRS, Alice can compute it without even knowing y.

Bob treats these matrices as LWE public key matrices, and, for each $i = 1, \ldots, n$, he generates an LWE sample $s^T(A_i + y_i G) + e_i$. He then sends these LWE samples over to Alice. Since each A_i is uniform, the matrix $A_i + y_i G$ is again going to be random, so it is not going to reveal any information about y.

Alice then stacks all of these LWE samples and multiplies the result by $H_{f,y}$ on the right to get

$$(s^T[A_1 + y_1 G \parallel \cdots \parallel A_n + y_n G] + [e_1 \ \cdots \ e_n]) \cdot H_{f,y} = s^T(A_f + f(x,y) \cdot G) + \tilde{e},$$

where $\tilde{e} = [e_1 \ \cdots \ e_n] \cdot H_{f,y}$ is again a low norm vector, because $H_{f,y}$ has low-norm by construction.

Now, Alice can do something very similar to how we decrypted GSW. As in the GSW, assume the last component of s is 1. Then the last column of the matrix Alice computed gives:

$$s^T a_f + f(x,y) \cdot q/2 + \bar{e},$$

where a_f is the last column of A_f and $\bar{e}$ is the last column of $\tilde{e}$.

To recover $f(x,y)$ from this, we can modify the protocol so Bob also sends a rounding of $s^T a_f$ to Alice along with the LWE samples. Then Alice rounds her result as well and retrieves $f(x,y)$ by XORing the two; i.e., she does

$$\lceil s^T a_f + f(x,y) \cdot q/2 + \bar{e} \rfloor \oplus \lceil s^T a_f \rfloor$$

There are actually multiple problems with this protocol. First, A_f could leak information about Alice's input x. However, there's a bigger problem for security of Bob. The matrix $H_{f,y}$ depends on y, so unless Alice knows what y is, she cannot compute $H_{f,y}$ by herself. Thus, even correctness would require Bob to send y to Alice, defeating the purpose of the protocol.

Step 2. We now modify the protocol to protect Bob's input.

In this protocol, Alice instead uses the homomorphic version $f' = \text{FHE-EVAL}(f[x], \cdot)$ and sends $A_{f'}$ to Bob. Then, Bob computes an encryption $ct = \text{FHE-ENC}(y)$ and also forms LWE samples $s^T(A + ct \otimes G) + e^T$ which he sends to Alice along with ct and $\lceil s^T a_{f'} \rfloor$.

Having received the ciphertext ct, Alice can now compute the matrix $H_{f',ct}$ securely. Repeating the previous steps yields

$$(s^T(A + ct \otimes G) + e^T) \cdot H_{f',ct} \approx s^T(A_{f'} + f'(ct) \otimes G)$$

and, from this and $\lceil s^T a_{f'} \rfloor$, she can output f' evaluated on ct, i.e.,

$$f'(ct) = \text{FHE-EVAL}(f[x], \text{FHE-ENC}(y)) = \text{FHE-ENC}(f(x,y))$$

In the previous construction, at this stage, the output $f(x,y)$ appeared in the clear. In the current setting, however, Alice only obtains $\text{FHE-ENC}(f(x,y))$, which is unusable for correctness, since she does not possess the FHE secret key to retrieve $f(x,y)$.

To recover $f(x,y)$, we therefore somehow need a method to do FHE decryption within Key-Equation, but without actually revealing any information about the FHE secret key, as that would compromise Bob's security.

In order to achieve this, we will make use of two crucial facts:

1. FHE decryption is nearly linear in the secret key; i.e. $\text{FHE-DEC}(sk, ct) \approx \langle ct, st \rangle$, and
2. There is asymmetry in the multiplication of Key-Equation.

To see the latter, recall that for multiplying two bits in the Key-Equation, we used a low norm matrix of the form

$$[H_{g_1,y} \parallel H_{g_2,y}] \cdot \begin{bmatrix} -G^{-1}(A_{g_2}) \\ g_1(y) \cdot I \end{bmatrix}$$

Note that even though we are computing the multiplication of $g_1(y)$ and $g_2(y)$, the construction of $H_{f,y}$ in the Key-Equation requires knowledge of only one of the inputs, namely $g_1(y)$. Furthermore, for addition gates, we actually do not need to know either $g_1(y)$ or $g_2(y)$, since we simply take $H_{f,y} = H_{g_1,y} + H_{g_2,y}$. Hence, this means that for an inner product between x and y, doing the Key-Equation requires knowledge of only one of the two inputs. In other words, we can do the Key-Equation of $f(x, y) = \langle x, y \rangle$ even when we only know one of the inputs, say x. This is because for the multiplications $x_i y_i$, we only need to know one of x_i or y_i, and for the addition, we do not need to know either.

Now, suppose we take $x = \text{FHE-ENC}(f(x, y))$ and $y = sk$ to be the secret key of this FHE encryption. Then, by leveraging the Key-Equation asymmetry structure, we can effectively compute the inner product between the two, i.e. $\langle \text{FHE-ENC}(f(x, y)), sk \rangle \approx \text{FHE-DEC}(sk, ct)$, without explicit knowledge of the secret key. This is exactly what we are going to use to get to our next step.

Step 3. In this step, we use the above observation to fix the correctness problem of step 2.

In this protocol, let B be another matrix in the CRS. Now, Bob computes $s^T(B + sk \otimes G) + e'$ and sends it to Alice along with ct and $\lceil s^T a' \rfloor$. Note that since B is random, $B + sk \otimes G$ is also uniformly random, so by the LWE assumption, this does not reveal any information about sk.

Having received ct and knowing $f[x]$, Alice can obtain the encryption $\text{FHE-ENC}(f(x, y))$ using the publicly computable FHE evaluation. Then, using the asymmetry property of the Key-Equation, she computes the Key-Equation matrix H' that encodes the inner product $\langle \text{FHE-ENC}(f(x, y)), sk \rangle$ using only her knowledge of $\text{FHE-ENC}(f(x, y))$.

Following the same steps as before, Alice then computes

$$(s^T(A + ct \otimes G) + e^T \| s^T(B + sk \otimes G) + e'^T) \cdot H' \approx s^T(A' + \langle \text{FHE-ENC}(f(x, y)), sk \rangle G)$$

Looking at the last column of this, Alice obtains something of the form

$$s^T a' + f(x, y) \cdot q/2 + \bar{e} + e',$$

where a' is the last column of A', and $\bar{e}$, e' are low-norm vectors we get from LWE and FHE.
Finally, at this stage, Alice can round this result and XOR it with $\lceil s^T a' \rfloor$, which she received from Bob, to obtain $f(x, y)$ in the clear.

The key takeaway here is that we can actually do FHE decryption under Key-Equation without knowledge of the FHE secret. This is a very powerful tool.

Security of Bob (sketch). The fact that ct and $s^T(B + sk \otimes G) + e^T$ do not reveal anything about y and sk follows from the security of FHE encryption and the LWE-assumption, respectively. To see why sending $\lceil s^T a' \rfloor$ in the clear does not compromise Bob's security, we note that we can compute Bob's share from Alice's share and $f(x, y)$ alone. Specifically, a simulator can compute Alice's share, round it, and then set Bob's share to be the XOR of $f(x, y)$ and Alice's share. Then, this would match the real distribution of $\lceil s^T a' \rfloor$ except with negligible probability, showing that Bob's message leaks no additional information about y or the FHE secret key.

Step 4. In the final step, we modify the protocol to also protect Alice's input.
Using the following simple modification, we can actually achieve statistical protection of Alice's input.

Instead of directly computing f, we define a related function g that takes (x, r) as Alice's input and $(y, 0)$ as Bob's input, and computes

$$g((x, r), (y, 0)) = f(x, y) + \langle 0, r \rangle = f(x, y)$$

Thus, g produces the same output as f, but operates on a slightly larger input domain. The inclusion of the random vector r is what we need to ensure input hiding for Alice.

As before, Alice computes the matrix A_f corresponding to f. We also introduce a set of matrices $\{D_i\}$ corresponding to the r- and 0-components of g. Alice then computes $A_f + \sum r_i D_i$ and sends it to Bob.

Bob encodes the LWE samples for f as before, but now also encodes the zeroes using $s^T(D_i)$ (plus the error). Using these encodings and the gate structure, Alice computes

$$s^T(A_f + f(x, y) \cdot G) + s^T(\sum r_i D_i) = s^T(A_f + \sum r_i D_i) + f(x, y) \cdot G$$

Following the same steps as before, Alice can then recover $f(x, y)$ in the clear.

Security of Alice (sketch). To ensure Alice's security, we need to show that $\sum_i r_i D_i$ is statistically close to uniform, so that Alice's message reveals nothing about A_f. This actually follows from the Leftover Hash Lemma (LHL). Note that r_i's are binary values. Representing the matrices D_i as columns of a larger matrix D, we can write $\sum_i r_i D_i = r^T D$. If there are sufficiently many D_i's, then by LHL, $\sum_i r_i D_i$ is statistically close to uniform. Hence, the message sent by Alice hides A_f, ensuring her security.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. *Laconic function evaluation and applications.* Cryptology ePrint Archive, Paper 2018/409, 2018.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, Dhinakaran Vinayagamurthy. *Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits.* Eurocrypt 2014.