Notes for Lecture 15

1 Last Time

Last time we discussed fully homomorphic encryption (FHE). Today, we will continue the topic by showing:
- The security of the FHE protocol discussed in Lecture 14.
- A bootstrapping result which boosts an FHE handling a limited number of operations to an FHE handling an unlimited number of operations. We need this boost as the scheme discussed produces noise of size $\mathsf{poly}(n)^T$ after computation of depth T, which limits the feasible depth of computation.

2 On FHE

Recall the scheme from last time for encrypting a binary message x :

$$\mathsf{Gen}() : \mathsf{sk} = \begin{pmatrix} s \\ -1 \end{pmatrix} \text{ where } s \xleftarrow{\$} \mathbb{Z}_q^{n-1}.$$
$$\mathsf{pk} = \begin{pmatrix} P \\ s^T P + e^T \end{pmatrix} \text{ where } P \xleftarrow{\$} \mathbb{Z}_q^{(n-1)\times m} \text{ for } m = n\lceil \log(q) \rceil,$$
e is short discrete Gaussian noise.

$\mathsf{Enc}(\mathsf{pk}, x) : C = \mathsf{pk} \cdot R + x \cdot G$, where
$R \xleftarrow{\$} \{0,1\}^{m\times m}$,
$$G = \begin{pmatrix} 1 & 2 & 4 & 8 & ... & & & & \\ & & & & 1 & 2 & 4 & 8... & \\ & & & & & & & & \ddots \end{pmatrix} \in \mathbb{Z}_q^{m \times n\lceil \log q \rceil}.$$

$\mathsf{Dec}(sk, C)$: Compute $\mathsf{sk}^T C = x \cdot \mathsf{sk}^T \cdot G - e^T$. By the structure of G, there exists a component where $\mathsf{sk}^T \cdot G$ is large, but e is small. Thus, a simple rounding enables differentiating between $x = 0$ and $x = 1$.

2.1 Security

We already proved the fully homomorphic (additive and multiplicative) property of the scheme. Now, we will prove CPA security via a hybrid argument. Consider:
• $\mathbf{H}_0$: Encrypt $m = 0$ according to the protocol.
• $\mathbf{H}_1$: Encrypt $m = 0$ according to the protocol, with the only difference that we switch to $\mathsf{pk} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$.
• $\mathbf{H}_2$: Encrypt $m = 1$ according to the protocol, with the only difference that we switch to $\mathsf{pk} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$.
• $\mathbf{H}_3$: Encrypt $m = 1$ according to the protocol.

Suppose, for the sake of contradiction, that there exists a PPT adversary A which breaks CPA security. According to the CPA definition (see Lecture 1), as the message space is given by $\{0, 1\}$, it must be the case that:

$$|Pr[1 \longleftarrow A(\mathbf{H}_0)] - Pr[1 \longleftarrow A(\mathbf{H}_3)]| \geq \epsilon(n)$$

for some non-negligible ϵ. By the usual triangle inequality, this means that:

$$\begin{aligned}&|Pr[1 \longleftarrow A(\mathbf{H}_0)] - Pr[1 \longleftarrow A(\mathbf{H}_1)]|+\\ &|Pr[1 \longleftarrow A(\mathbf{H}_1)] - Pr[1 \longleftarrow A(\mathbf{H}_2)]|+\\ &|Pr[1 \longleftarrow A(\mathbf{H}_2)] - Pr[1 \longleftarrow A(\mathbf{H}_3)]| \geq \epsilon(\lambda),\end{aligned}$$

so at least one of the three expression should be non-negligible. However:
• $|Pr[1 \longleftarrow A(\mathbf{H}_0)] - Pr[1 \longleftarrow A(\mathbf{H}_1)]|$ and $|Pr[1 \longleftarrow A(\mathbf{H}_2)] - Pr[1 \longleftarrow A(\mathbf{H}_3)]|$ are negligible by LWE security (see Lecture 9).
• $|Pr[1 \longleftarrow A(\mathbf{H}_1)] - Pr[1 \longleftarrow A(\mathbf{H}_2)]|$ is also negligible. The intuition behind this fact is that when m is large, $P \cdot R$ is statistically indistinguishable from a uniformly random matrix. This, however, implies that $P \cdot R + 0 \times G$ and $P \cdot R + 1 \times G$ are both statistically indistinguishable from a random matrix and, therefore, from each other. This argument is made precise through the following lemma appearing in the original paper for FHE from LWE [BV11, p.9]:

Theorem 2.1 (Matrix-Vector Leftover Hash Lemma): *Let* $\kappa \in \mathbb{N}, n \in \mathbb{N}, q \in \mathbb{N}$, *and* $m > n \log q + 2\kappa$. *Let* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}, \mathbf{r} \xleftarrow{\$} \{0, 1\}^m, \mathbf{y} \xleftarrow{\$} \{0, 1\}^n$. *Then,*

$$\Delta\Big((\mathbf{A}, \mathbf{A}^T\mathbf{r}), (\mathbf{A}, \mathbf{y})\Big) \leq 2^{-\kappa},$$

where Δ *is the statistical difference between the distributions.*[1]

[1]Statistical difference denotes the advantage any adversary (including computationally unbounded ones) can make in distinguishing the distributions. Up to a factor of 2, it is true that $\Delta(D_0, D_1) = \sum_x |Pr[x|D_0] - Pr[x|D_1]|$ for discrete distributions.

One can extend this result to $\mathbf{r} \xleftarrow{\$} \{0,1\}^{m \times m}$ as in our set-up.

As all three differences are negligible, this leads to a contradiction. The protocol is CPA secure as desired.

2.2 Bootstrapping

We move on to bootstrapping. We will ignore the particulars of the scheme and discuss a very general bootstrapping result. The goal is roughly to take an FHE that can only handle limited computation and turn into into an FHE that can handle arbitrary computation. For this, two assumptions are necessary.

2.2.1 Assumptions

Assumption 1, Circular Security: The original FHE remains secure even if given $\mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$ (see [BV11, Definition 3.8] for a precise definition). Circular security arises as a natural requirement in the bootstrapping scheme, but is also of independent interest related to other applications.
We will assume circular security for the FHE protocol. This property, however, is only conjectured. While it is known how to build schemes satisfying circular security from from LWE (see for example [App+09]), this is an open problem in the case of FHE schemes. Still, evidence points that existing FHE schemes should be circularly secure.

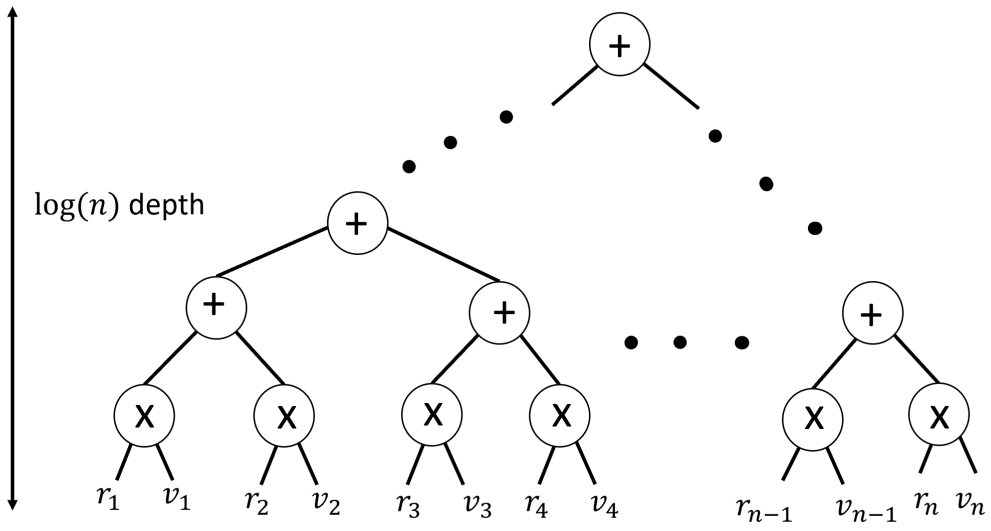
Assumption 2: The scheme can homomorphically evaluate its own decryption function. To make this statement precise, recall that the decryption function can be written as a (boolean) circuit. We require that our scheme supports evaluations at least as complex as the scheme's decryption.
We can prove this assumption for our scheme. In our scheme on page 1, decryption takes the form of taking an inner product followed by rounding. As the scheme is limited by depth[2] of computation, we need to show that it supports enough depth of computation to perform these two operations:

- Rounding can be done easily with $\log(n)-$depth circuits as one only needs to ensure that higher-order bits (which are $O(\log(n))$ in number) are non-zero.

- Taking a modular inner product can also be done with a computation of depth $\log(n)$. Recall the usual formula: $\langle r, v \rangle = \sum_{i=1}^{n} r_i v_i \pmod{q}$. We can compute this via a shallow circuit in the following way. First, we ignore the

[2]For other schemes, the fundamental limiting factor and, therefore, benchmark for evaluating its own decryption might be different from depth.

modular arithmetic and perform the following circuit of depth $\log(n)$, in which we multiply at the bottom level and add on each of the others (as on the figure below). Then, we can also easily reduce the result modulo q with another $\log(n)-$depth circuit.



By setting parameter large enough, our FHE scheme can handle depth $\log n$.
Note: For all known FHE schemes based on lattices, Assumption 2 holds.

2.3 The Bootstrapping Procedure

Having established the two assumptions of our FHE scheme, we will ignore most of its details and present a generic bootstrapping technique.

Suppose $c_1 = \mathsf{Enc}(\mathsf{pk}, m_1), c_2 = \mathsf{Enc}(\mathsf{pk}, m_2)$. We want to compute

$$c = \mathsf{Enc}(\mathsf{pk}, m_1 \oplus m_2),$$

where $\oplus$ is an arbitrary binary operation that we will regard to be XOR for concreteness. To compute c, we first define the following function:

$$D_{c_1,c_2}(\mathsf{sk}) := \mathsf{Dec}(\mathsf{sk}, c_1) \oplus \mathsf{Dec}(\mathsf{sk}, c_2).$$

Note that anyone can compute[3] D_{c_1,c_2} as it requires knowledge of Dec, c_1 and c_2. Of

[3]Here, we actually need a slightly stronger assumption than Assumption 2. We want to be able two evaluate two instances of decryption rather than one. While this might be of practical importance, the asymptotic behaviour does not change and our scheme still satisfies this assumption.

course, one cannot evaluate it without knowledge of sk. Having access to u[4] and D_{c_1,c_2}, however, we can homomorphically evaluate $D_{c_1,c_2}(u)$. This will give $\mathsf{Enc}(\mathsf{pk}, D_{c_1,c_2}(u))$ by the definition of FHE. On the other hand,

$$\mathsf{Enc}(\mathsf{pk}, D_{c_1,c_2}(\mathsf{sk})) = \mathsf{Enc}(\mathsf{pk}, \mathsf{Dec}(\mathsf{sk}, c_1) \oplus \mathsf{Dec}(\mathsf{sk}, c_2)) =$$
$$= \mathsf{Enc}(\mathsf{pk}, \mathsf{Dec}(\mathsf{sk}, c_1)) \oplus \mathsf{Enc}(\mathsf{pk}, \mathsf{Dec}(\mathsf{sk}, c_2)) = \mathsf{Enc}(m_1 \oplus m_2) = c,$$

as desired.

Notes:

- The result c might be very noisy. If applying Dec twice in the homomorphic evaluation of $D_{c_1,c_2}(u)$ requires the maximum depth that the FHE scheme can handle, no further homomorphic computation with c might be possible. This, however, is not a real limitation as the bootstrapping procedure only applies homomorphic evaluations on u. Even if c_1 and c_2 are already very noisy, as long as u is not, the bootstrapping technique can be applied.

- The bootstrapping technique can similarly be applied to "refresh" a ciphertext $c = \mathsf{Enc}(\mathsf{pk}, m)$ if the computation has reached its maximum depth. Defining $D_c(x) := \mathsf{Dec}(x, c)$, one can homomorphically evaluate $D_c(u)$ to obtain $\mathsf{Enc}(\mathsf{pk}, \mathsf{Dec}(\mathsf{sk}, c)) = \mathsf{Enc}(\mathsf{pk}, m)$. This results in a new decryption of m, which will only be at depth equal to the depth of the decryption function. In other words, one can use this technique to occasionally construct new instances of the ciphertext at a lower fixed depth equal to $depth(\mathsf{Dec})$.

- The above "refreshing" technique leads to fast implementations of FHE. They work by performing FHE operations according to the original protocol and "refreshing" ciphertext instances only when necessary. Other optimizations have also been developed. Currently, the overhead (due to bootstrapping) has been reduced to an amortized $\approx 10^6$ steps per homomorphic operation. While this is far from FHE computation on real complex data, it still allows up to 1000 FHE operations per second which enables some non-trivial real-world computation.

-*Question: Doesn't this procedure require that the decryption function only performs operations such as addition and multiplication?*
A: Yes. However, additional and multiplication are enough for all bitwise operations as they generate AND, OR, and NOT :

$$x \text{ AND } y = x \times y$$
$$\text{NOT } x = 1 - x$$
$$x \text{ OR } y = x + y - x \times y$$

[4]As access to $u = \mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$ is crucial to the scheme, we needed assumption 1 which states that knowing u does not break security.

Having finished the details of the FHE protocol, we will move on to some generalizations of concepts previously introduced in the class.

3 Attribute-Based Encryption

First, we will generalize IBE - identity based encryption (see Lecture 11). Recall that in IBE we wanted to encrypt to any single user. Sometimes in practice, however, this is not sufficient. For example, we may want to encrypt to all students or all students in the CS department. In other words, there can be a more general access structure in which you don't encrypt to a single user, but instead to all users who satisfy a certain property. This idea gives rise to attribute-based encryption.

Attribute-Based Encryption Syntax:

$$(\mathsf{mpk}, \mathsf{msk}) \longleftarrow \mathsf{Gen}(1^\lambda)$$

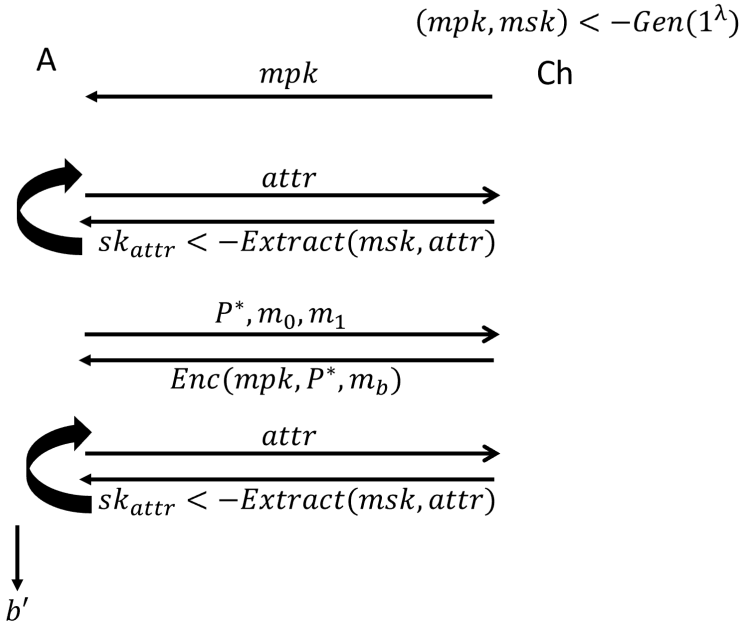
$$\mathsf{sk}_{attr} \longleftarrow \mathsf{Extract}(\mathsf{mpk}, attr)$$

$$c \longleftarrow \mathsf{Enc}(\mathsf{mpk}, P, m)$$

$$m' \longleftarrow \mathsf{Dec}(\mathsf{sk}_{attr}, c),$$
where $m' = m$ if $P(attr) = 1$,
and m is hidden if $P(attr) = 0$.

Above, mpk and msk are the master public key and master secret key; $attr$ corresponds to an attribute that characterizes a group of users; the policy P is a function which accepts only the group of users characterized by $attr$.
The decryption condition states that only users who hold the secret key for an attribute $attr$ accepted by policy P can decrypt a ciphertext encrypted for this policy.

We define security in a similar way to IBE security. We run an experiment EXP_b between an adversary A and a challenger Ch. In this experiment, the challenger Ch has a hidden bit b. Then, the challenger generates a public key - secret key pair $(\mathsf{mpk}, \mathsf{msk})$ and sends mpk to the adversary A. Then, A queries as many times as she wishes an attribute $attr$ and receives the secret key sk_{attr}. At some point of time, A sends a policy P^* and two messages m_0 and m_1 and the challenger encrypts m_b according to P^*. After querying for more secret keys sk_{attr}, the adversary A makes a guess b' for b. A wins if $b = b'$. Of course, we require that $P^*(attr) = 0$ for all attributes $attr$ queried by A in the experiment as otherwise the task is trivial. Graphically:



For security, we require that for each PPT adversary A :

$$\left| Pr[1 \longleftarrow A(\text{EXP}_0)] - Pr[1 \longleftarrow A(\text{EXP}_1)] \right| \leq \mathsf{negl}(\lambda)$$

Note: In the special case when the policies accept only a single user id, i.e. are of the form $P(x) = (x == id)$, this is the definition of IBE.

Example: Consider the example of two users who can collude. For example u_1 is a student in the math department and u_2 is a professor in CS. Then, neither user is allowed for a policy $P := (student) \wedge (CS\ department)$. For security, we require that even if u_1 and u_2 collude they cannot satisfy the policy even though u_1 satisfies the first condition and u_2 the second.
In practice, we often encode properties of users by attribute lists. For instance, to formalize the above example in the language of attributes, we can encode:

Encoding Properties in an Attribute List				
User	Student	Professor	Math	CS
u_1	1	0	1	0
u_2	0	1	0	1

Then, the example policy will ask for the and operation between the bits for student and cs, i.e. $P = (Student)$ AND (CS).

Notes:

- Many of the original schemes worked in this way by computing the "and" of a certain subset of bits. Today, we also have schemes that can handle attributes of arbitrary length and can perform arbitrary circuits (beyond "and") on them.

- *Question: What attribute lengths are supported* - $\log(\lambda)$ *or* $\mathsf{poly}(\lambda)$?
A: We usually think of attributes having a fixed $\mathsf{poly}(\lambda)$ length. We do need $\mathsf{poly}(\lambda)$ length so that we are able to support IBE as a special case (see Lecture 12). It is also possible to define attributes of variable length, but this requires policies to be defined as Turing machines rather than circuits.

- *Question: Do we need a poly-sized message space?*
A: No. We just require attributes to be of polynomial length[5].

4 Functional Encryption

A closely related concept to attribute-based encryption is functional encryption. We motivate it through the following example.

Example: As a client of a mail server, I have a key pair $(\mathsf{pk}, \mathsf{sk})$ with which I decrypt incoming mail. The server receives a message $c = \mathsf{Enc}(\mathsf{pk}, m)$. One desirable property of the server is to apply a spam-filter SF and decide if the incoming mail is spam and, if so, discard it before sending it to me. How can the mail server run SF without learning the content?

Idea based on FHE: One idea is to resolve this by using FHE encryption. If we homomorphically apply SF to c, the server gets $SF(c) = \mathsf{Enc}(\mathsf{pk}, SF(m))$. This, however, doesn't help the mail server as the result of the spam filter is hidden under the encryption. Thus, the mail server cannot learn $SF(m)$ (unless we give the server sk, but we do not necessarily trust the server to view the contents of our messages). Thus, the only thing the mail server can do is send us $\mathsf{Enc}(\mathsf{pk}, SF(m))$. However,it cannot discard spam before forwarding it to us.

We get the following intuition from this example. The problem of the FHE approach is that we can only give an encrypted version of $SF(m)$. However, we want the server to learn $SF(m)$ but "nothing else."

[5]This means that the attribute space can be of size $2^{\mathsf{poly}(\lambda)}$

To formalize this requirement, we first need to introduce the syntax of functional encryption:

$$
\begin{aligned}
(\mathsf{mpk}, \mathsf{msk}) &\longleftarrow \mathsf{Gen}(1^\lambda) \\
\mathsf{sk}_f &\longleftarrow \mathsf{Extract}(\mathsf{msk}, f), \text{ where } f \text{ is a function.} \\
c &\longleftarrow \mathsf{Enc}(\mathsf{mpk}, m) \\
x &\longleftarrow \mathsf{Dec}(\mathsf{sk}_f, c).
\end{aligned}
$$

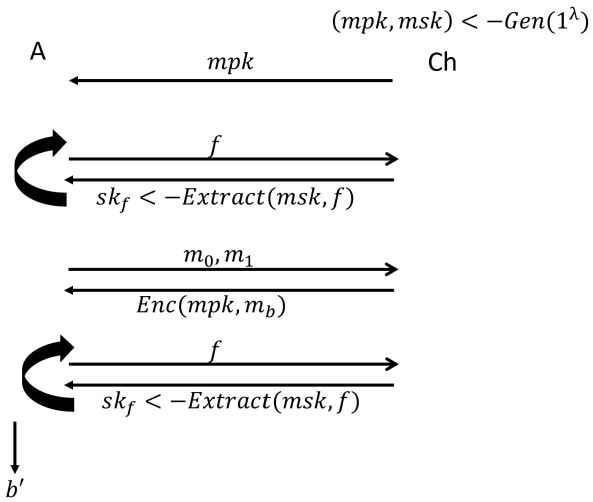
Above, f is the function to be applied - for instance SF in the mail server.
The correctness requirement is given by $f(m) = \mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{mpk}, m))$ with probability 1. For security, we (informally) require that sk_f reveals nothing about m but $f(m)$.

We relate this syntax to the mail server instance as follows. We give the server sk_{SF}. By decrypting $\mathsf{Dec}(\mathsf{sk}_{SF}, c)$, the spam filter learns $SF(m)$ but nothing more about the message. The spam filter can now discard spam messages m for which $SF(m) = 1$.

-Question: Are collusions allowed?
A: In general, we want collusion-resistance for the same reasons as in IBE and traitor-tracing. If we issue several different secret keys, we don't want different serves to be able to collude and get access to encrypted information.

We will discuss two approaches to security. The first resembles the definition of attribute-based encryption security:

Definition 4.1 (Security of Functional Encryption): We describe the experiment through a picture:



Here, the requirement on queries is that $f(m_0) = f(m_1)$ for all queried functions f as otherwise A's task is trivial. Again, if we denote this experiment by EXP_b, we want

CPA-style security. That is, for all PPT adversaries A,

$$\left| Pr[1 \longleftarrow A(\text{EXP}_0)] - Pr[1 \longleftarrow A(\text{EXP}_1)] \right| \leq \mathsf{negl}(\lambda).$$

Unfortunately this definition is not ideal. Suppose that F is an OWF and A gets sk_F. Now, suppose that one encrypts $c \longleftarrow \mathsf{Enc}(\mathsf{mpk}, x)$ for a random x. The adversary learns $F(x)$ by the definition of functional encryption. Ideally, A should learn "nothing else," including x. Unfortunately, the above security definition doesn't justify this. It is possible that the functional encryption is secure with respect to Definition 4.1 and A is still able to recover x.

Thus, one can attempt a different definition which formalizes "learns nothing else" through a simulator as in Zero-Knowledge (see Lecture 9)[6]. Unfortunately, it turns out that it is impossible to obtain a functional encryption scheme satisfying this notion of security. We only provide intuition for this. Suppose that one wants to encrypt a PRG $F : \{0,1\}^\delta \longrightarrow \{0,1\}^{\delta + s(\delta)}$. We want A to learn the output $F(m) = \mathsf{Dec}(\mathsf{sk}_F, c)$ but nothing else. Thus, a simulator that outputs the result of F should look random (as a PPT adversary A cannot distinguish a PRG output from random). However, if the ciphertext c is much smaller than the PRG output in size, $|c| << \delta + s(\delta)$, then $\mathsf{Dec}(\mathsf{sk}_F, c)$ will be a distribution over only $2^{|c|}$ numbers and, thus, will have an entropy much lower than that of the uniformly random distribution over $\{0,1\}^{\delta + s(\delta)}$. Thus, the output of the PRG $\mathsf{Dec}(\mathsf{sk}_F, c)$ could actually be computationally distinguishable from a random string, which is undesirable.

As a consequence, we usually use the indistinguishability definition (Definition 4.1). However, we need to be careful as the notion of security it guarantees is limited.

References

[App+09] Benny Applebaum et al. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems". In: vol. 5677. Aug. 2009, pp. 595–618. ISBN: 978-3-642-03355-1. DOI: 10.1007/978-3-642-03356-8_35.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. "Functional Encryption: Definitions and Challenges". In: May 2011, pp. 253–273. ISBN: 978-3-642-19570-9. DOI: 10.1007/978-3-642-19571-6_16.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient Fully Homomorphic Encryption from (Standard) LWE". In: vol. 2011. Oct. 2011, pp. 97–106. DOI: 10.1109/FOCS.2011.12.

[6] See, for example, Definition 4 in [BSW11] for such a definition for functional encryption