Oskar Goldhahn

A Look Into Homomorphic Cryptography and the BV Homomorphic Encryption Scheme

Bachelor's project in Mathematical Sciences
Supervisor: Kristian Gjøsteen
May 2020

Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Oskar Goldhahn

A Look Into Homomorphic Cryptography and the BV Homomorphic Encryption Scheme

Bachelor’s project in Mathematical Sciences
Supervisor: Kristian Gjøsteen
May 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Kunnskap for en bedre verden

Abstract

We take a look at the BV homomorphic encryption scheme and security, correctness, and compactness proofs for this encryption scheme as an example of how homomorphic encryption is done, and how proofs in the field are written. We expand the proofs in the original paper in an attempt to make the arguments easier to follow.

Contents

Chapter 1

Introduction

The purpose of homomorphic encryption is to allow computation on encrypted data without the need to decrypt first. This allows us to outsource computation without trust. This has numerous applications in, for instance, cloud computing and encrypted databases.

A conventional encrypted database can only respond to data requests or do limited restructuring of the data. With a database built upon homomorphic encryption we can do data analysis on the database directly.

The idea was first brought up by Rivest, Adleman and Dertouzos in 1978[6]. It turns out that several encryption schemes allow some computation on them, though this might not always be intentional. RSA, for instance, allows us to multiply encrypted numbers.

The first encryption scheme that allows us to evaluate any binary circuit on encrypted data was published by Gentry in 2009[4]. His paper includes a method called bootstrapping, that extends the set of circuits an existing encryption scheme can evaluate. After his paper there has been increased interest, and a number of papers have been written on the topic of homomorphic encryption.

In this thesis we will investigate homomorphic encryption by looking at a scheme developed by Brakerski and Vaikuntanathan[3]. The scheme is based on the DLWE[1] problem, that asks us to distinguish between inner products with an added noise term, and uniformly sampled numbers. We will display a proof that the decryption of the scheme gives us the desired result and a proof regarding the security of the scheme. Finally we will take a short look at possible improvements to the scheme.

[1]Decision learning with errors

Chapter 2

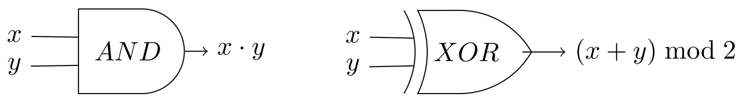
Theory

Here we will display some basic definitions and theorems of homomorphic encryption and some that will help us with proofs later.

The cryptographic definitions are taken from the paper on the BV Scheme[3], but modified to fit the more modern language of *A Guide to Fully Homomorphic Encryption*[1].

2.1 Homomorphic encryption

A homomorphic encryption scheme is a regular encryption scheme endowed with the additional property of evaluation, the ability to compute with ciphertexts. In this thesis we will formalize computation using binary circuits made up from AND and XOR gates. By appending these



gates to each other without cycles we can compute any Boolean function, and by using binary representations of numbers and multiple circuits we can compute any function between finite sets of integers, since the set of gates is universal. It might seem like the restriction to functions between finite sets prevents us from doing arbitrary computation, but by looking at the number of input bytes we can bound the size of the input and chose a circuit that can handle inputs below that bound. The gates correspond to addition and multiplication modulo 2, which gives us a very useful parallel to algebra.

Definition 2.1.1 ($\mathcal{C}$-Evaluation Scheme[1, 3])**.** Let $\mathcal{C}$ be a set of binary circuits. A $\mathcal{C}$-Evaluation Scheme is a quadruple of probabilistic polynomial time algorithms as follows

- **Key generation**. The algorithm $\mathsf{KGen}(1^\kappa)$ that takes a unary representation of the security parameter κ and outputs a public encryption key pk, a public evaluation key evk and a secret decryption key sk.
- **Encryption**. The algorithm $\mathsf{Enc}_{\mathsf{pk}}(\mu)$ takes a public encryption key pk, a single bit of plaintext $\mu \in \{0, 1\}$ and outputs a ciphertext c.
- **Homomorphic evaluation**. The algorithm $\mathsf{Eval}_{\mathsf{evk}}(C, c_1, \dots, c_n)$ takes an evaluation key evk, a circuit $C \in \mathcal{C}$ and a tuple of ciphertexts of length equal to the number of inputs to the circuit. It outputs a ciphertext c.
- **Decryption**. The algorithm $\mathsf{Dec}_{\mathsf{sk}}(c)$ takes a ciphertext c and a secret decryption key sk and outputs a single bit of plaintext μ.

This is essentially the same as the definition of public key encryption schemes, only with the addition of an algorithm that turns circuits and ciphertext into ciphertext. This gives us a framework for computing on ciphertexts, but is not enough by itself. As stated, Eval could return encryptions of 1 on every input, which would not be very useful. We want the algorithm to actually use the circuit to do something. This motivates the following definition.

Definition 2.1.2 (Correct Evaluation[1, 3])**.** A $\mathcal{C}$-evaluation scheme has correct evaluation if for any $C \in \mathcal{C}$ and any set of plaintext bits $\mu_1, \ldots, \mu_n$ with size corresponding to the input of C we have

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{evk}}(C, \mathsf{Enc}_{\mathsf{pk}}(\mu_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(\mu_n))) \neq C(\mu_1, \ldots, \mu_n)] = \mathsf{negl}(\kappa)$$

In other words, correct evaluation means encrypting, evaluating and then decrypting with overwhelming probability[1] gives the same result as passing the plaintexts through the circuit. With this property we can already evaluate circuits on encrypted data, but there is a little edge case to take care of first. Consider an evaluation scheme where the evaluation algorithm returns its inputs and the circuit, and the decryption algorithm decrypts the ciphertexts, passes them into the circuit and returns the output. In this case the evaluation does little work. Most of the work is done in the decryption, which defeats the purpose of homomorphic encryption. The solution here is to limit the size of the output of Eval. If we limit the size enough we can not fit the circuit anymore.

Definition 2.1.3 (Compactness[1])**.** A $\mathcal{C}$-evaluation scheme is compact if there is a fixed polynomial p such that for any key-triple $(\mathsf{sk}, \mathsf{pk}, \mathsf{evk})$ output by $\mathsf{KGen}(1^\kappa)$ and valid input to Eval the size of the output is bounded by $p(\kappa)$ bits.

So far we have allowed $\mathcal{C}$ to be an arbitrary set of circuits, but if this is small we might not be able to do the computation we want, even if the circuit needed is very small. This motivates the following definitions

Definition 2.1.4 (Fully homomorphic[1])**.** A $\mathcal{C}$-evaluation scheme is fully homomorphic if it has correct evaluation, is compact, and $\mathcal{C}$ is the set of all binary circuits.

Definition 2.1.5 (Leveled fully homomorphic[1])**.** A $\mathcal{C}$-evaluation scheme is leveled homomorphic if KGen takes an additional input L which specifies the maximum depth of the circuits that can be evaluated. Further requirements are correct evaluation and compactness for circuits with depth less than L and that the length of the evaluation output does not depend on L. If $\mathcal{C}$ is the set of all binary circuits we say that the scheme is leveled fully homomorphic.

The depth of a circuit is taken to be the maximal number of gates from an input to an output[2]. Using a leveled homomorphic encryption scheme we can evaluate any circuit that is smaller than our bound L. For most practical purposes this is enough, but we gain a bit more flexibility from using a fully homomorphic encryption scheme since we then do not need a bound of the circuit depth before we generate the keys and start encrypting.

2.2 Security

So far we have only talked about the functionality of encryption schemes, but there is also the question of security. How do we make it hard to decrypt messages without the secret key? To aid in this goal we first have to formalize what it means for a homomorphic encryption scheme to be secure. Secure from what exactly?

[1] $\mathsf{negl}(n)$ is the class of functions whose inverse grows faster than any polynomial, but the takeaway is that the probability is miniscule. The reason we allow a small probability of failure is mostly to simplify proofs.

[2] By padding with multiplication by 1 or addition by 0 we can make every path have the same number of gates.

Definition 2.2.1 (Adversaries)**.** A distinguishing adversary is a probabilistic algorithm that takes a sample from a distribution as input and outputs 0 or 1.

The purpose of the adversary here is to try to tell the difference between two distributions, for example the distribution of encryptions of 0 and 1. We measure the performance of an adversary as follows

Definition 2.2.2 (Advantage)**.** Let X and Y be random variables. We define the advantage of an adversary $\mathcal{A}$ distinguishing between X and Y as follows

$$\mathsf{Adv}_{(X,Y)}[\mathcal{A}] \overset{\text{def}}{=} |\Pr[\mathcal{A}[X] = 1] - \Pr[\mathcal{A}[Y] = 1]|$$

We use the term hybrid for a pair of random variables we calculate the advantage on.[3]

An adversary with high advantage is able to tell which distribution samples come from with great certainty, while an adversary with low advantage is barely better than a guess. We get the following definition for security

Definition 2.2.3 (IND-CPA security[3])**.** We say that a homomorphic encryption scheme HE is (t, ϵ)-IND-CPA secure if for any adversary $\mathcal{A}$ that runs in time t it holds that

$$\mathsf{Adv}_{\text{CPA}}[\mathcal{A}] \overset{\text{def}}{=} \Big|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{evk}, \mathsf{BV.Enc}_{\mathsf{pk}}(1)) = 1] - \Pr[\mathcal{A}(\mathsf{pk}, \mathsf{evk}, \mathsf{BV.Enc}_{\mathsf{pk}}(0)) = 1]\Big| \leq \epsilon$$

where pk and evk are obtained from $\mathsf{HE.KGen}$.

It should be noted that the adversary has access to the public key and the evaluation key, which means that it can generate as many encryptions as it wants and also homomorphically evaluate them to gain a decryptable ciphertext.

This definition of security formalizes what it means for an efficiently computable adversary to be able to tell the difference between encryptions of 1 and 0 (or not). It turns out that this makes it hard to guess even for an adversary with additional information, for example that ones are more likely than zeros. If an adversary could guess more precisely if the distribution of ones and zeros is skewed, then it might as well skew it itself by pretending to get an encryption of 1 half the time, but in reality output 1 and 0 with equal probability.

2.3 Tools

When we introduce the scheme we study in this paper we will need a few additional theorems for the security proof. Adversaries and advantage formalize what it means to be able to compute the difference between two distributions, but as always there are stronger notions than computation.

Definition 2.3.1 (Statistical distance[7])**.** The statistical distance between random variables X and Y taking values in the finite set S is defined as

$$\Delta[X, Y] \overset{\text{def}}{=} \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$$

One way to think about this is as the non-overlapping regions of the distributions.

Theorem 2.3.2. [7] *Let X, Y and Z be random variables such that Z is independent from the other two, then*

$$\Delta[X, Y] = \Delta[(X, Z), (Y, Z)]$$

[3]Hybrids can be more complicated, but this is enough for our purpose.

Spreading out both of the distributions in the same way will make them overlap neither more nor less.

Theorem 2.3.3. [7] *Let S and T be finite sets, X and Y random variables with support in S, and $f : S \to T$ be a function, then*

$$\Delta[f(X), f(Y)] \leq \Delta[X, Y]$$

Running the overlapping regions through a function will not separate them.

The same result also holds for probabilistic algorithms between finite sets, not just functions, since we can always add an independent random variable as input to a function instead of using it internally in the algorithm and adding independent random variables do not change the statistical distance.

We also need the universal hash lemma, which relies on the following definitions

Definition 2.3.4 (Universal hashing[7])**.** Let X be a uniform random variable over the finite set I. A family of functions $\{f_i\}_{i \in I}$ between finite sets S and T is called a universal family of hash functions if

$$\Pr[f_X(s) = f_X(s')] \leq \frac{1}{|T|}$$

for all $s, s' \in A$ with $s \neq s'$.

Definition 2.3.5 (Collision probability[7])**.** The collision probability of a random variable X taking values in the finite set S is

$$\sum_{s \in S} \Pr[X = s]^2$$

Theorem 2.3.6 (Leftover hash lemma[7])**.** *Let $\{f_i\}_{i \in I}$ be a universal family of hash functions from S to T. Let X, Y and Z be independent random variables, where X is uniform over I, Z is uniform over T and Y takes values in S. If β is the collision probability of Y, then*

$$\Delta[(X, f_X(Y)), (X, Z)] \leq \frac{1}{2}\sqrt{|S|\,\beta}$$

The security of encryption schemes rely on a given difficult problem in computer science. The scheme we study in this thesis uses the DLWE problem, that asks us to tell the difference between noisy inner products and uniformly sampled numbers. Without the noise this is a very easy problem that boils down to solving a system of linear equations, but once noise is added it becomes very difficult.

Definition 2.3.7 (DLWE[3, 5])**.** Let q and k be integers, χ be a distribution over $\mathbb{Z}_q$ and $\mathbf{s}$ be a vector in $\mathbb{Z}_q^n$. The probabilistic algorithm $A_{\mathbf{s},\chi}$ chooses a vector $\mathbf{a} \leftarrow\!\!\$\ \mathbb{Z}_q^n$ uniformly at random, samples $e \leftarrow\!\!\$\ \chi$ and outputs $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q \times \mathbb{Z}_q^n$. The goal of the $\mathsf{DLWE}_{n,q,\chi}$-problem is to distinguish between this algorithm and an algorithm that outputs uniform samples from $\mathbb{Z}_q \times \mathbb{Z}_q^n$, given access to q, k, χ and infinitely many outputs from the algorithms.

This is the decision version of the learning with errors problem that asks us to find $\mathbf{s}$. The infinite set of outputs correspond to being given access to the algorithm itself without being able to look at the inside. Such an algorithm that gives the adversary access to small pieces of information is often called an oracle.

There are some guarantees for the difficulty of the problem, but for that we require an additional definition.

Definition 2.3.8 (B-bounded[3])**.** We call a countable set of random variables $\{X_i\}_{i \in \mathbb{N}}$ with values in the integers B-bounded if

$$\Pr[X_i > B] \leq 2^{\tilde{\Omega}(i)}$$

For a given $B \geq n$ and weak conditions on q we can find a B-bounded distribution such that often difficult and extensively studied lattice problems can be reduced to $\mathsf{DLWE}_{n,q,\chi}$.[3, 5]

This gives the problem a bit firmer footing as a foundation for cryptography. As for why we want to bound the size of the noise, we will see in the security proof.

2.4 Bootstrapping

Making a leveled fully homomorphic encryption scheme is a tall order, but there are tools to help us get there. The first fully homomorphic encryption scheme[4] used a technique called bootstrapping that allows us to turn some $\mathcal{C}$-evaluation schemes into a leveled fully homomorphic encryption scheme. To explain the bootstrapping process we need the following definition

Definition 2.4.1 (Bootstrappable[3, 4])**.** A $\mathcal{C}$-evaluation scheme is called bootstrappable if it has correct evaluation, the decryption algorithm can be represented as a circuit and

$$\mathsf{BV.Dec}_{\mathsf{sk}}(c)\ AND\ \mathsf{BV.Dec}_{\mathsf{sk}}(c') \quad \text{and} \quad \mathsf{BV.Dec}_{\mathsf{sk}}(c)\ XOR\ \mathsf{BV.Dec}_{\mathsf{sk}}(c') \tag{2.1}$$

as circuits with inputs sk, c and c' are both in $\mathcal{C}$.

In other words, we want the scheme to correctly evaluate its own decryption in addition to an arbitrary logic gate.

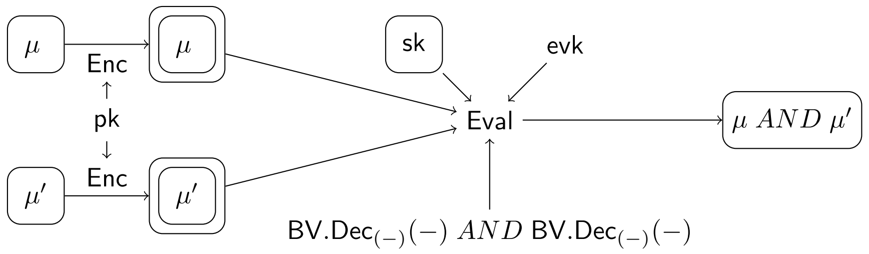


Figure 2.1: Bootstrapping

The boxes are encryption layers.

Bootstrapping works by encrypting the ciphertexts a second time[4], and then running Eval with an encrypted secret key and the circuit in (2.1). Essentially we decrypt the inner layer of the doubly encrypted ciphertexts and then use a logic gate afterwards. We can do the same with the XOR gate, and by appending them together we can evaluate any circuit we want. It should be noted that the number of encryptions of keys we provide limit the size of the circuits we can evaluate, only making a leveled homomorphic encryption scheme.[5]

The important point is that correct evaluation only guarantees that we will get what we want if we use a newly encrypted ciphertext, but since we encrypt right before we evaluate when bootstrapping this is the case. When building large circuits this way, the probability of a failed decryption increases, but it is never more than a sum of the probabilities for failure in each gate. We can keep this small by using different sets of keys for each layer generated by different security parameters. This way we can use a small security parameter for the final gate to get a small output, but use larger for the others to keep the chance of failed decryption at bay. Using more than one set of keys helps security as well.

This way of making leveled fully homomorphic encryption schemes is wasteful. Most of the computing power is spent on computing encryptions and decryptions, not the actual logic gates.

[4]Since the evaluator is doing the encryption, it can chose all the randomness in the way it wants so that the encryptions are trivial and possibly look the same as the original plaintext.

[5]With additional requirements for security we can also use it to make a fully homomorphic encryption scheme by having the secret keys encrypted in a loop instead of a chain.

This can be optimized somewhat if we can evaluate more than one gate in addition to the decryption circuit, but only to an extent.

As a final note on bootstrapping, we address security. The only extra tool the adversary gets in the bootstrapped encryption scheme that we do not get in the original scheme, is a sequence of encryptions of secret keys such that each is encrypted by the public key corresponding to the next secret key. Being able to decrypt the message would mean that we could extract some additional information from the encryption of these keys, but the last key is decrypted under a completely unknown key, so it is hard to distinguish between this and a randomly sampled one. Likewise it is hard to distinguish between the final ciphertext and an encryption of a random bit. This argument continues until we see that it is hard to distinguish all the ciphertexts and encrypted secret keys from random ones, so it is also hard to see the difference between an encryption of one and an encryption of zero.

A formal proof would address how hard it is to distinguish by finding a bound for the advantage, as we will do in the security proof for the BV encryption scheme.

Theorem 2.4.2 (Bootstrapping theorem[4]). *Given a bootstrappable $\mathcal{C}$-evaluation scheme we can construct a leveled fully homomorphic encryption scheme.*

Chapter 3

BV Scheme

The scheme we will investigate was introduced by Brakerski and Vaikuntanathan[3].

3.1 Overview

The BV encryption scheme is built upon the DLWE problem, with the ciphertexts having the following structure

$$(\mathbf{a}, \mu + 2e + \langle \mathbf{a}, \mathbf{s} \rangle) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

Where μ is the plaintext, $e \leftarrow_{\$} \chi$ is the noise from the DLWE problem, $\mathbf{a}$ is any vector, and $\mathbf{s}$ is part of the secret key. Decryption is just subtracting the inner product from the right side of the ciphertext and taking the remainder when dividing by two. The difficult part is obtaining the homomorphic properties. Addition already works by adding the ciphertexts, but if we try to multiply we end up with products of inner products, which has non-linear terms over the secret key and does not let us decrypt using the inner product, which only has linear terms. The solution here is to make the ciphertext linear in the secret key again by using encryptions of products $\mathbf{s}[i]\mathbf{s}[j]$ and $\mathbf{s}$ to replace the non-linear terms over the old secret key with linear terms over a new secret key. This can be done many times to multiply without introducing non-linear terms.

In addition we reduce the size of the decryption circuit by changing the modulus and dimension of the final ciphertext from q and n to p and k to facilitate bootstrapping. When doing this we also need a different distribution $\hat{\chi}$ to sample the noise from.

The encryption scheme will be able to evaluate circuits with depth $2L$, such that every even layer only has multiplication (AND) gates and every odd layer has only addition (XOR) gates. We require all signals to go through one gate on every layer. A circuit that does not meet this final requirement can be padded by multiplying by 1 or adding 0 if necessary.

3.2 Scheme description

- Key generation $\mathsf{BV.KGen}(1^\kappa)$[1]: Sample $L+1$ vectors $\mathbf{s}_0, \ldots, \mathbf{s}_L \leftarrow_{\$} \mathbb{Z}_q^n$, $\mathbf{a}_{\ell,i,j,\tau} \leftarrow_{\$} \mathbb{Z}_q^n$ and $e_{\ell,i,j,\tau} \leftarrow_{\$} \chi$. Define $\mathbf{s}_{\ell-1}[0] \overset{\text{def}}{=} 1$ and compute, for all $\ell \in [L]$, $0 \leq i \leq j \leq n$, and $\tau \in \{0, \ldots, \lfloor \lg q \rfloor\}$, the value

$$\psi_{\ell,i,j,\tau} := \left(\mathbf{a}_{\ell,i,j,\tau}, b_{\ell,i,j,\tau} := \langle \mathbf{a}_{\ell,i,j,\tau}, \mathbf{s}_\ell \rangle + 2 \cdot e_{\ell,i,j,\tau} + 2^\tau \cdot \mathbf{s}_{\ell-1}[i] \cdot \mathbf{s}_{\ell-1}[j] \right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

[1]The parameters depend on κ.

Define $\Psi \overset{\text{def}}{=} \{\psi_{\ell,i,j,\tau}\}_{\ell,i,j,\tau}$.

Choose random matrix $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$ and a vector $\mathbf{e} \leftarrow_\$ \chi^m$, and compute $\mathbf{b} \coloneqq \mathbf{A}\mathbf{s}_0 + 2\mathbf{e}$.

Sample $\hat{\mathbf{s}} \leftarrow_\$ \mathbb{Z}_p^k$, $\hat{\mathbf{a}}_{i,\tau} \leftarrow_\$ \mathbb{Z}_p^k$ and $\hat{e}_{i,\tau} \leftarrow_\$ \hat{\chi}$ and compute for all $i \in [n]$ and $\tau \in \{0, \dots, \lfloor \lg q \rfloor\}$, the value

$$\hat{\psi}_{i,\tau} \coloneqq \left(\hat{\mathbf{a}}_{i,\tau}, \hat{b}_{i,\tau} \coloneqq \langle \hat{\mathbf{a}}_{i,\tau}, \hat{\mathbf{s}}\rangle + \hat{e}_{i,\tau} + \left\lfloor \frac{p}{q} \cdot (2^\tau \cdot \mathbf{s}_L[i]) \right\rceil \mod p\right) \in \mathbb{Z}_p^k \times \mathbb{Z}_p.$$

Define $\hat{\Psi} \overset{\text{def}}{=} \{\hat{\psi}_{i,\tau}\}_{i,\tau}$.

The output is the secret key $\mathsf{sk} = \hat{\mathbf{s}}$, the evaluation key $\mathsf{evk} = (\Psi, \hat{\Psi})$, and the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$.

- Encryption $\mathsf{BV.Enc}_{\mathsf{pk}}(\mu)$: $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$, $\mu \in \mathbb{Z}_2$. Sample a vector $\mathbf{r} \leftarrow_\$ \{0,1\}^m$ and set

$$\mathbf{v} \coloneqq \mathbf{A}^T\mathbf{r} \quad \text{and} \quad w \coloneqq \langle \mathbf{b}, \mathbf{r}\rangle + \mu$$

  Output is the ciphertext $c \coloneqq (\mathbf{v}, w, 0)$.

- Homomorphic evaluation $\mathsf{BV.Eval}_{\mathsf{evk}}(C, c_1, \dots, c_t)$ where C should be layered with only one type of gates on each layer, odd layers being addition and even layers being multiplication. There should be exactly $2L$ layers. Homomorphic addition of $c = (\mathbf{v}, w, \ell)$ and $c' = (\mathbf{v}', w', \ell)$ gives $c'' = (\mathbf{v} + \mathbf{v}', w + w', \ell)$.

  For homomorphic multiplication of $c = (\mathbf{v}, w, \ell)$ and $c' = (\mathbf{v}', w', \ell)$ define $\mathbf{v}[0] \overset{\text{def}}{=} -w$ and $\mathbf{v}'[0] \overset{\text{def}}{=} -w'$. We compute $h_{i,j} \coloneqq \mathbf{v}[i] \cdot \mathbf{v}'[j]$ and find a binary representation using bits $h_{i,j,\tau}$ such that $h_{i,j} = \sum_{\tau=0}^{\lfloor \lg q\rfloor} h_{i,j,\tau} \cdot 2^\tau$. Using $\psi_{\ell,i,j,\tau} = (\mathbf{a}_{\ell,i,j,\tau}, b_{\ell,i,j,\tau})$ we finally get the output of the homomorphic multiplication

$$\left(\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor \lg q\rfloor\}}} h_{i,j,\tau} \cdot \mathbf{a}_{\ell+1,i,j,\tau}, \quad \sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor \lg q\rfloor\}}} h_{i,j,\tau} \cdot b_{\ell+1,i,j,\tau}, \quad \ell + 1\right) \tag{3.1}$$

  We use these homomorphic operations to iterate through C until we end up with a single ciphertext $c = (\mathbf{v}, w, L)$.

  For this ciphertext compute $h_i \coloneqq \frac{q+1}{2}\mathbf{v}[i] \mod p$ and its binary representation $h_{i,\tau}$ such that $h_i = \sum_{\tau=0}^{\lfloor \lg p\rfloor} h_{i,j,\tau} \cdot 2^\tau$. Finally, we use $\hat{\psi}_{i,\tau} = (\hat{\mathbf{a}}_{i,\tau}, \hat{b}_{i,\tau})$ to compute

$$(\hat{\mathbf{v}}, \hat{w}) \coloneqq \left(2 \cdot \sum_{\substack{i\in[0,n]\\ \tau\in\{0,\dots,\lfloor \lg p\rfloor\}}} h_{i,\tau} \cdot \hat{\mathbf{a}}_{i,\tau}, \quad 2 \cdot \sum_{\substack{i\in[0,n]\\ \tau\in\{0,\dots,\lfloor \lg p\rfloor\}}} h_{i,\tau} \cdot \hat{b}_{i,\tau}\right) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$$

  The output of the evaluation is $\hat{c} \coloneqq (\hat{\mathbf{v}}, \hat{w})$.

- Decryption $\mathsf{BV.Dec}_{\mathsf{sk}}(\hat{c})$: We decrypt $(\hat{\mathbf{v}}, \hat{w})$ as follows

$$\mu' \coloneqq ((\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}}\rangle) \bmod p) \bmod 2$$

  where $w' = (w \bmod p)$ is the number in $[-p/2, p/2)$ such that $w \equiv w' \pmod p$. The output is μ'.

3.3 Correctness

An evaluation scheme would not be of much use without correct evaluation. We show this by bounding the size of the noise. The proof is an extended version of the proof by Brakerski and Vaikuntanathan[3].

To obtain correctness we need to restrict a few of the parameters. We restrict the distributions χ and $\hat{\chi}$ to be B- and $\hat{B}$-bounded respectively to at least bound the initial noise. As we will see setting $B = n$ and $\hat{B} = k$ gives us the properties we need. We set $m = \lfloor (n+1) \lg q + 2\kappa + 2 \rfloor$ to get a bound on the size of a newly encrypted ciphertext. We require both moduli to be odd with $p \geq 16nk \lg 2q$ and $q \leq 2^{n^\epsilon}$ for some fixed $\epsilon \in (0,1)$. Finally we need both n and k to be polynomial in the the security parameter κ. We do this by choosing a fixed positive integer c and setting $k = \kappa$ and $n = k^c$.

Theorem 3.3.1. *For fixed ϵ, we can find $L \in \Omega(\epsilon \lg n)$ such that* BV *has correct evaluation for large enough parameters.*

Proof. For the purpose of simplifying the algebra we will always be assuming that the integer parameters are greater than 10. We start by establishing an equality for the ciphertexts that have been produced through encryption and partial evaluation $c = (\mathbf{v}, w, \ell)$.

$$w - \langle \mathbf{v}, \mathbf{s}_\ell \rangle = \mu_c + 2e_c \mod q \tag{3.2}$$

where μ_c is the bit value that we would have gotten if we had done all the homomorphic operations we did to get c on the original plaintexts instead and e_c is a small noise term. Everything is done modulo q. We proceed by induction.

For the base case recall that $\mathbf{v} = \mathbf{A}^T\mathbf{r}$, $w = \langle \mathbf{b}, \mathbf{r} \rangle + \mu$ and $\mathbf{b} = \mathbf{A}\mathbf{s}_0 + 2\mathbf{e}$. Since we have done no homomorphic operations $\mu_c = \mu$. $w - \langle \mathbf{v}, \mathbf{s}_\ell \rangle = \langle \mathbf{A}\mathbf{s}_0 + 2\mathbf{e}, \mathbf{r} \rangle + \mu - \langle \mathbf{A}^T\mathbf{r}, \mathbf{s}_0 \rangle = \langle \mathbf{A}\mathbf{s}_0 + 2\mathbf{e}, \mathbf{r} \rangle + \mu - \langle \mathbf{A}\mathbf{s}_0, \mathbf{r} \rangle = \mu + 2\langle \mathbf{e}, \mathbf{r} \rangle$. Since $\mathbf{r}$ was sampled from $\{0,1\}^m$ we get $\langle \mathbf{e}, \mathbf{r} \rangle \leq |\mathbf{e}| \leq m \max_{1 \leq i \leq m}(|\mathbf{e}_i|)$.

Now we show that if $c = (\mathbf{v}, w, \ell)$ and $c' = (\mathbf{v}', w', \ell)$ both obey the equality, then so will the result of their homomorphic addition $(\mathbf{v} + \mathbf{v}', w + w', \ell)$. $w + w' - \langle \mathbf{v} + \mathbf{v}', \mathbf{s}_\ell \rangle = w - \langle \mathbf{v}, \mathbf{s}_\ell \rangle + w' - \langle \mathbf{v}', \mathbf{s}_\ell \rangle = \mu_c + 2e_c + \mu_{c'} + 2e_{c'} = \mu_c + \mu_{c'} + 2(e_c + e_{c'})$, which is of the form of equation (3.2) with the noise being at most the sum of the noises of the summands plus one.

Now we show the same for homomorphic multiplication of c and c'. Recall the multiplication computation (3.1), which gives the left side of equation (3.2) as

$$\begin{aligned}
&\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau} b_{\ell+1,i,j,\tau} - \left\langle \sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau}\mathbf{a}_{\ell+1,i,j,\tau},\quad \mathbf{s}_{\ell+1}\right\rangle\\
&\quad= \sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau}\left(\langle \mathbf{a}_{\ell+1,i,j,\tau},\mathbf{s}_{\ell+1}\rangle + 2e_{\ell+1,i,j,\tau} + 2^{\tau}\mathbf{s}_\ell[i]\mathbf{s}_\ell[j]\right) - \left\langle \sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau}\mathbf{a}_{\ell+1,i,j,\tau},\quad \mathbf{s}_{\ell+1}\right\rangle\\
&\quad= \sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau}(2e_{\ell+1,i,j,\tau} + 2^{\tau}\mathbf{s}_\ell[i]\mathbf{s}_\ell[j])\\
&\quad= 2\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau}e_{\ell+1,i,j,\tau} + \sum_{0\le i\le j\le n} h_{i,j}\mathbf{s}_\ell[i]\mathbf{s}_\ell[j]\\
&\quad= 2\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} h_{i,j,\tau}e_{\ell+1,i,j,\tau} + \sum_{0\le i\le j\le n} \mathbf{v}[i]\mathbf{v}'[j]\mathbf{s}_\ell[i]\mathbf{s}_\ell[j]\\
&\quad= 2\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} (h_{i,j,\tau}e_{\ell+1,i,j,\tau}) + ww' - \sum_{j\in[1,n]}(\mathbf{v}[j]w'\mathbf{s}_\ell[j] + w\mathbf{v}'[j]\mathbf{s}_\ell[j]) + \sum_{1\le i\le j\le n}\mathbf{v}[i]\mathbf{v}'[j]\mathbf{s}_\ell[i]\mathbf{s}_\ell[j]\\
&\quad= 2\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} (h_{i,j,\tau}e_{\ell+1,i,j,\tau}) + (\mu_c + 2e_c + \langle\mathbf{v},\mathbf{s}_\ell\rangle)(\mu_{c'} + 2e_{c'} + \langle\mathbf{v}',\mathbf{s}_\ell\rangle)\\
&\qquad - \langle\mathbf{v},\mathbf{s}_\ell\rangle(\mu_{c'} + 2e_{c'} + \langle\mathbf{v}',\mathbf{s}_\ell\rangle) - \langle\mathbf{v}',\mathbf{s}_\ell\rangle(\mu_c + 2e_c + \langle\mathbf{v},\mathbf{s}_\ell\rangle) + \langle\mathbf{v},\mathbf{s}_\ell\rangle\langle\mathbf{v}',\mathbf{s}_\ell\rangle\\
&\quad= 2\sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}} (h_{i,j,\tau}e_{\ell+1,i,j,\tau}) + (\mu_c + 2e_c)(\mu_{c'} + 2e_{c'})\\
&\quad= 2\left(e_c\mu_{c'} + e_{c'}\mu_c + 2e_ce_{c'} + \sum_{\substack{0\le i\le j\le n\\ \tau\in\{0,\dots,\lfloor\lg q\rfloor\}}}(h_{i,j,\tau}e_{\ell+1,i,j,\tau})\right) + \mu_c\mu_{c'}
\end{aligned}$$

This is in the form of equation (3.2).

Finally we show that a similar equality

$$\hat{w} - \langle\hat{\mathbf{v}},\hat{\mathbf{s}}\rangle = \mu_{\hat{c}} + 2\hat{e} \mod p$$

holds for ciphertexts of the form $\hat{c} = (\hat{\mathbf{v}}, \hat{w})$ that have been fully evaluated. We again start with the left side. All calculation is done mod p.

$$\begin{aligned}
\frac{p+1}{2}(\hat{w} - \langle\hat{\mathbf{v}},\hat{\mathbf{s}}\rangle) &= \Big(\sum_{\substack{i\in[0,n]\\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}\hat{b}_{i,\tau} - \langle\sum_{\substack{i\in[0,n]\\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}\hat{\mathbf{a}}_{i,\tau},\hat{\mathbf{s}}\rangle\Big)\\
&= \sum_{\substack{i\in[0,n]\\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{b}_{i,\tau} - \langle\hat{\mathbf{a}}_{i,\tau},\hat{\mathbf{s}}\rangle)\\
&= \sum_{\substack{i\in[0,n]\\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}\left(\hat{e}_{i,\tau} + \left\lfloor \frac{p}{q}2^{\tau}\mathbf{s}_L[i]\right\rceil\right)
\end{aligned}$$

We get rid of the rounding by adding another noise term $|e| \leq \frac{1}{2}$.

$$\begin{aligned}
\sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + \left\lfloor \frac{p}{q} 2^\tau \mathbf{s}_L[i] \right\rceil) &= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e + \frac{p}{q} 2^\tau \mathbf{s}_L[i]) \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \sum_{i\in[0,n]} h_i \mathbf{s}_L[i] \frac{p}{q} \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \sum_{i\in[0,n]} \frac{q+1}{2} \mathbf{v}[i] \mathbf{s}_L[i] \frac{p}{q} \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \frac{q+1}{2} \frac{p}{q} (\langle \mathbf{v}, \mathbf{s}_L \rangle - w) \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \frac{p(q+1)}{2q} (\mu_c + 2e_c + Nq) \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \frac{p(q+1)}{2q} \mu_c + \frac{p(q+1)}{q} e_c \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \frac{p(q+1)}{2q} \mu_c + \frac{p}{q} e_c \\
&= \sum_{\substack{i\in[0,n] \\ \tau\in\{0,\dots,\lfloor\lg p\rfloor\}}} h_{i,\tau}(\hat{e}_{i,\tau} + e) + \frac{p+1}{2} \mu_c + (\frac{p}{q} - 1) \frac{\mu_c}{2} + \frac{p}{q} e_c
\end{aligned}$$

Clearly $\frac{p+1}{2}\mu_c$ is an integer, which means that the sum of the rest of the terms is also an integer. Multiplying $\frac{p+1}{2}\mu_c$ by 2 yields $\mu_c \bmod p$. Retrieving the expression we started with, gives

$$2\frac{p+1}{2}(\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) = \hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu_c + 2\hat{e} \mod p \tag{3.3}$$

where

$$\hat{e} = \sum_{i,\tau} h_{i,\tau}(\hat{e}_{i,\tau} + e) + (\frac{p}{q} - 1)\frac{\mu_c}{2} + \frac{p}{q} e_c.$$

The problem now, is to find out when the noise is small enough for ciphertexts to be decrypted correctly. We want the final noise term plus evaluated plaintext to be less than $\frac{p}{2}$ in magnitude. We define a function on a partly evaluated ciphertext $c = (\mathbf{v}, w, \ell)$ as follows

$$\eta(c) \overset{\text{def}}{=} |\mu_c + 2e_c|$$

This is simply the magnitude of the invariant in (3.2) without taking the modulus. This is not well defined, since the noise is not uniquely defined by the invariant, so we use the noise that minimizes the function value. Since χ and $\hat{\chi}$ are B and $\hat{B}$-bounded respectively, with all but negligible probability the polynomial in n and k samples will all be less than the bounds[2]. From now on we disregard that negligible case. Let η_ℓ be the maximal size of η for any of the ciphertexts evaluated up to and including multiplicative layer ℓ and $\hat{\eta}$ be the same for completely evaluated ciphertexts.

[2]This is why we need n and k to be polynomial in κ.

Our earlier work gives $\eta_0 \leq 2Bm + 1$ and the inequality

$$\eta_\ell \leq 2\eta_{\ell-1}^2 + 2B\frac{(n+1)(n+2)}{2}(\lg q + 1) \leq 2\max\{2\eta_{\ell-1}, (n+2)\sqrt{B \lg 2q}\}^2$$

Using $m > (n+1)\lg q + 2\kappa$ we can bound the noise of freshly encrypted ciphers as follows

$$\eta_0 \leq 2Bm + 1 = \max\{2Bm+1, (n+2)\sqrt{B\lg 2q}\},$$

which recursively gives us a bound for every multiplicative layer including the last one.

$$\eta_\ell \leq 8\eta_{\ell-1}^2 \leq \frac{(8\eta_0)^{2^\ell}}{8} \leq \frac{(16Bm+8)^{2^\ell}}{8}$$

Using this and (3.3) with its bounds we get

$$\hat{\eta} \leq (\lg 2p)(n+1)(2\hat{B}+1) + \frac{p}{q}\eta_L + 2 \leq 4\lg(2p)n\hat{B} + \frac{p}{q}\eta_L$$

From the parameters we get

$$\frac{p}{4} \geq 4\lg(2p)n\hat{B}$$

It remains to show that

$$\frac{p}{4} > \frac{p}{q}\eta_L \Leftrightarrow \frac{1}{4} > \frac{1}{q}\eta_L$$

Let us insert the parameters

$$\frac{\eta_L}{q} \leq \frac{(16Bm+8)^{2^L}}{8q} \leq \frac{(16n(n+1)n^\epsilon + 32\kappa + 32 + 8)^{2^L}}{8 \cdot 2^{n^\epsilon}} \leq \frac{(32n^{2+\epsilon})^{2^L}}{8\cdot 2^{n^\epsilon}}$$

If L is a small enough fraction of $\epsilon \lg n$, then the denominator is asymptotically larger than the numerator. This means that for large enough n the fraction is always smaller than $\frac{1}{4}$ and we have our result. □

3.4 Compactness

When bootstrapping we get compactness for free, but compactness is still useful in case we want to use the system without bootstrapping.

Theorem 3.4.1. *The* BV *evaluation scheme is compact*

Proof. The output of BV.Eval is an element of $\mathbb{Z}_p^k \times \mathbb{Z}_p$. These elements can be represented using $(k+1)\lceil \lg p \rceil$ bits. Since $k = \kappa$ and $p = \lceil 16\kappa^{c+1}\kappa^{c\epsilon} \rceil$ are both polynomial in κ the result follows. □

3.5 Security

Even if correct, an evaluation scheme is not very useful if it can be exploited by adversaries. We show that BV is secure on the condition that certain DLWE problems are hard. The proof is an expanded version of the proof by Brakerski and Vaikuntanathan[3].

Our proof is a hybrid argument. We start with an adversary that gets the evaluation key, the public key and an encryption of 0 or 1. We gradually change the information we give the adversary until everything is uniform and for each change we show that the difference between the advantages of the original and the changed distribution is bounded. In the final hybrid everything is uniform, so the advantage is zero and we can sum up the differences in the advantages to get a bound on the advantage of an adversary that gets all the information.

Theorem 3.5.1. *Let n, k, q, p, L and κ be parameters as described in the scheme. Let χ and $\hat{\chi}$ be distributions over the integers as described in the scheme. If $\mathsf{DLWE}_{n,q,\chi}$ and $\mathsf{DLWE}_{k,p,\hat{\chi}}$ are both (t,ϵ)-hard, then the scheme is $(t - \mathsf{poly}(\kappa), 2(L+1)(2^{-\kappa}+\epsilon))$-IND-CPA.*

Proof. Let $\mathcal{A}$ be an IND-CPA adversary for BV that runs in time t'. We proceed by a sequence of hybrids.

- **Hybrid** $\hat{H}$: The adversary gets pk and evk distributed as if generated by $\mathsf{BV.KGen}$ and an encryption of 0 for one random variable and 1 for the other encrypted with $\mathsf{BV.Enc}$. This hybrid is the same as the situation in IND-CPA, so

$$\mathsf{Adv}_{\hat{H}}[\mathcal{A}] = \mathsf{Adv}_{\text{CPA}}[\mathcal{A}] \overset{\text{def}}{=} \delta$$

- **Hybrid** H_{L+1}: Same as previous hybrid except that $\hat{\Psi}$ in the evaluation key is sampled uniformly instead of being computed properly.

  An adversary $\hat{\mathcal{B}}$ would then be able to solve the $\mathsf{DLWE}_{k,p,\hat{\chi}}$ problem in time $t' + \mathsf{poly}(\kappa)$ with advantage

$$\mathsf{Adv}_{\mathsf{DLWE}_{k,p,\hat{\chi}}}[\hat{\mathcal{B}}] \geq \tfrac{1}{2}\left|\mathsf{Adv}_{H_{L+1}}[\mathcal{A}] - \mathsf{Adv}_{\hat{H}}[\mathcal{A}]\right|.$$

  The adversary generates Ψ, pk and all the $\mathbf{s}_0,\ldots,\mathbf{s}_L$ as in $\mathsf{BV.KGen}$, but generates $\hat{\Psi}$ as n pairs given by the oracle in the DLWE problem, only adding the third term, $\lfloor \frac{p}{q} \cdot (2^\tau \cdot \mathbf{s}_L[i])\rceil$, to the inner product (or uniformly sampled number), just like $\mathsf{BV.KGen}$ would. Then the adversary uniformly samples $\mu \leftarrow_{\$} \{0,1\}$. The adversary outputs 1 if $\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(\mu)) = \mu$ and 0 otherwise. Clearly the inputs fed to $\mathcal{A}$ correspond to either H_{L+1} or $\hat{H}$. Let P be the event that the first case holds.

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{DLWE}_{k,p,\hat{\chi}}}[\hat{\mathcal{B}}] &= \Big|\Pr\Big[\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(\mu)) = \mu \;\Big|\; \mathsf{P}\Big] \\
&\qquad - \Pr\Big[\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(\mu)) = \mu \;\Big|\; \overline{\mathsf{P}}\Big]\Big| \\
&= \tfrac{1}{2}\Big|\Pr\Big[\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(1)) = 1 \;\Big|\; \mathsf{P}\Big] + 1 \\
&\qquad - \Pr\Big[\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(0)) = 1 \;\Big|\; \mathsf{P}\Big] \\
&\qquad - \Pr\Big[\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(1)) = 1 \;\Big|\; \overline{\mathsf{P}}\Big] - 1 \\
&\qquad + \Pr\Big[\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(0)) = 1 \;\Big|\; \overline{\mathsf{P}}\Big]\Big| \\
&\geq \tfrac{1}{2}\left|\mathsf{Adv}_{H_{L+1}}[\mathcal{A}] - \mathsf{Adv}_{\hat{H}}[\mathcal{A}]\right|
\end{aligned}$$

  The additional work required by $\hat{\mathcal{B}}$ corresponds to the execution of (parts of) $\mathsf{BV.KGen}$, one uniform sampling from $\{0,1\}$ and a comparison. The amount of additional work is therefore polynomial in κ.

- **Hybrid** H_ℓ with $\ell \in [L]$: Same as $H_{\ell+1}$ except that $\psi_{\ell,i,j,\tau}$ is sampled uniformly instead of as in $\mathsf{BV.KGen}$. Similar to the last hybrid there is an adversary $\mathcal{B}_\ell$ that solves the $\mathsf{DLWE}_{n,q,\chi}$ with time $t' + \mathsf{poly}(\kappa)$ and advantage

$$\mathsf{Adv}_{\mathsf{DLWE}_{n,q,\chi}}[\mathcal{B}_\ell] \geq \tfrac{1}{2}\left|\mathsf{Adv}_{H_\ell}[\mathcal{A}] - \mathsf{Adv}_{H_{\ell+1}}[\mathcal{A}]\right|.$$

  The adversary generates the encryption key and $\mathbf{s}_0,\ldots,\mathbf{s}_{\ell-1}$ as usual and uses them to generate the corresponding parts of Ψ. The $\psi_{\ell,i,j,\tau}$ are the pairs given by the oracle multiplied by 2 and having $2^\tau \mathbf{s}_{\ell-1}[i][j]$ added to the inner product. Since the modulus is odd, multiplication by 2 keeps uniform distributions uniform. The rest of the evaluation key is sampled uniformly. Again the adversary samples $\mu \leftarrow_\$ \{0,1\}$ and outputs 1 if $\mathcal{A}(\mathsf{pk}, (\Psi, \hat{\Psi}), \mathsf{BV.Enc}(\mu)) = \mu$ and 0 otherwise. As in the previous proof we get one of two hybrids depending on what the oracle gives us and the algebra is similar.

In hybrid H_1 the entire evaluation key is uniform.

- **Hybrid** H_0: This hybrid is identical to H_1, except $\mathbf{b}$ is uniform instead of computed as $\mathbf{A}\mathbf{s}_0 + 2\mathbf{e}$.

  There exists an adversary $\mathcal{B}_0$ solving the $\mathsf{DLWE}_{n,q,\chi}$ with time $t' + \mathsf{poly}(\kappa)$ and advantage

  $$\mathsf{Adv}_{\mathsf{DLWE}_{n,q,\chi}}[\mathcal{B}_0] \geq \tfrac{1}{2}\left|\mathsf{Adv}_{H_0}[\mathcal{A}] - \mathsf{Adv}_{H_1}[\mathcal{A}]\right|.$$

  The adversary gets m samples from the oracle, multiplies them all by 2 and uses the result as $(\mathbf{A}, \mathbf{b})$. The rest of the adversary is similar to the two above and so is the algebra.

- **Hybrid** H_{rand}: Identical to H_0 except that the ciphertext is sampled uniformly instead of computed as $(\mathbf{A}^T\mathbf{r}, \langle \mathbf{b}, \mathbf{r}\rangle + \mu)$.

  Let $\mathbf{A}' \overset{\text{def}}{=} (\mathbf{A}, \mathbf{b})^T = \mathsf{pk}^T \leftarrow_\$ \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{y}' \overset{\text{def}}{=} (\mathbf{y}, y) \leftarrow_\$ \mathbb{Z}_q^{(n+1)}$ and $\mathbf{r} \leftarrow_\$ \{0,1\}^m$, be treated as uniform independent random variables. Treating $\mathbf{A}' \leftarrow_\$ \mathbb{Z}_q^{(n+1)\times m}$ as a function by matrix-vector multiplication, the family of functions is a universal family of hash functions from $\{0,1\}^m$ to $\mathbb{Z}_q^{(n+1)}$, so we can use the leftover hash lemma (2.3.6). First we need the collision probability of $\mathbf{r}$, which is 2^{-m} since it is uniform, then we need the size of the target space, which is q^{n+1}. Using $m > (n+1)\lg q + 2\kappa$ we get the bound

  $$\Delta[(\mathbf{A}', \mathbf{A}'\mathbf{r}), (\mathbf{A}', \mathbf{y})] \leq \tfrac{1}{2}\sqrt{2^{-m}q^{n+1}} < \tfrac{1}{2}\sqrt{2^{-2\kappa}} = \tfrac{1}{2}2^{-\kappa}.$$

  Using this statistical distance and the uniform independent random variables $\mu \leftarrow_\$ \{0,1\}$ and evk we can bound the difference between advantages of $\mathcal{A}$ on the last two hybrids.

$$\begin{aligned}
\Delta[(\mathbf{A}', \mathbf{A}'\mathbf{r}), (\mathbf{A}', \mathbf{y})] &= \Delta[(\mathbf{A}', \mathbf{A}'\mathbf{r}, \mu, \mathsf{evk}), (\mathbf{A}', \mathbf{y}, \mu, \mathsf{evk})] \\
&\geq \Delta\Big[\Big|\mathcal{A}[\mathsf{pk}, \mathsf{evk}, (\mathbf{A}^T\mathbf{r}, \langle \mathbf{b}, \mathbf{r}\rangle + \mu)] - \mu\Big|, \Big|\mathcal{A}[\mathsf{pk}, \mathsf{evk}, (\mathbf{y}, y + \mu)] - \mu\Big|\Big] \\
&\geq |\Pr[\mathcal{A}[H_0] = \mu] - \Pr[\mathcal{A}[H_{\mathsf{rand}}] = \mu]| \\
&= \tfrac{1}{2}\,|\Pr[\mathcal{A}[H_0] = 1 \mid \mu = 1] - \Pr[\mathcal{A}[H_0] = 1 \mid \mu = 0] \\
&\qquad - \Pr[\mathcal{A}[H_{\mathsf{rand}}] = 1 \mid \mu = 1] + \Pr[\mathcal{A}[H_{\mathsf{rand}}] = 1 \mid \mu = 0]| \\
&\geq \tfrac{1}{2}\left|\mathsf{Adv}_{H_0}[\mathcal{A}] - \mathsf{Adv}_{H_{\mathsf{rand}}}[\mathcal{A}]\right|
\end{aligned}$$

  In H_{rand} the public key, the evaluation key, the ciphertext and the message are all uniform and independent, so

  $$\mathsf{Adv}_{H_{\mathsf{rand}}}[\mathcal{A}] = 0$$

Adding up the differences we get

$$\mathsf{Adv}_{\hat{H}}[\mathcal{A}] \leq 2^{-\kappa} + 2(\mathsf{Adv}_{\mathsf{DLWE}_{k,p,\hat{\chi}}}[\hat{\mathcal{B}}] + \sum_{\ell=0}^{L} \mathsf{Adv}_{\mathsf{DLWE}_{k,p,\hat{\chi}}}[\mathcal{B}_\ell])$$

where $\mathcal{A}$ runs in time at least $t - \mathsf{poly}(\kappa)$, which gives the result. □

3.6 Bootstrapping

We will not display a complete proof of bootstrapability. The idea is to show that the circuit depth of the decryption circuit is asymptotically smaller than the depth of the circuits we can evaluate. To do this we need to take advantage of parallelization and efficient computational algorithms. A full proof can be found in the BV paper [3].

Chapter 4

Final remarks

Shortly after the BV scheme was published, Vaikuntanathan, Brakerski and Gentry wrote *Fully Homomorphic Encryption without Bootstrapping*[2], which presents a similar scheme with a number of improvements. The main improvement comes from the observation that the modulus switch that happens in the last step of BV.Eval actually reduces the size of the noise. It also reduces the modulus by a similar amount, which means that the noise to modulus ratio does not change much, but it turns out that it helps tremendously with keeping the noise in check when multiplying.

In the BV scheme the noise is approximately squared every time we multiply. Multiplying three times increases the noise from $2e$ to $\sim 2^8e^8$. If we divide the modulus by B every time we multiply, we instead get $\sim 2^8e^8/B^7$, which, if we chose $B \geq 2e$, is similar to the original noise, but the modulus is only reduced from q to $\sim q/B^3$. The ratio is a lot better than it is without changing modulus. We can use this method to keep the noise essentially constant while slowly whittling away at the modulus. Without modulus switching we get $\sim \lg \log_{2e}(q)$ multiplications before the noise becomes too large. With modulus switching we get $\sim \log_B(q)$, which grows exponentially faster than the other in the worst case.

Bibliography

[1] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192, 2015. http://eprint.iacr.org/2015/1192.

[2] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. http://eprint.iacr.org/2011/277.

[3] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Cryptology ePrint Archive, Report 2011/344, 2011. http://eprint.iacr.org/2011/344.

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

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

[6] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[7] Victor Shoup. *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, USA, 2 edition, 2009. https://shoup.net/ntb/ntb-v2.pdf.