(Guest Lecture) Fully Homomorphic Encryption

Keewoo Lee[1] *(UC Berkeley)*

[1] keewoole@gmail.com

April 29, 2025

In this note, we discuss the construction of a secret-key variant of the GSW[2] somewhat homomorphic encryption scheme.

[2] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013

GSW Scheme

In the same spirit as previous SHE constructions[3,4], we will begin with a ring homomorphism and introduce *noise* to achieve security, with the goal of constructing a somewhat ring-homomorphic encryption. Any somewhat ring-homomorphic encryption scheme suffices for our purpose, since the NAND gate can be simulated over any ring with unity using expression $1 - xy$ for $x, y \in \{0, 1\}$. (Note that the NAND gate alone forms a functionally complete[5] operator set.)

[3] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009

[4] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, 2010

[5] `https://en.wikipedia.org/wiki/Functional_completeness`

Ring Homomorphism from Eigenvalues

The starting point of the GSW scheme is a ring homomorphism defined by eigenvalues. More precisely, consider a scheme where a ciphertext encrypting a message $\mu \in \mathbb{Z}_q$ under a secret key $\boldsymbol{v} \in \mathbb{Z}_q^m$ is a matrix $\boldsymbol{C} \in \mathbb{Z}_q^{m \times m}$ satisfying:

$$\boldsymbol{C}\boldsymbol{v} = \mu \boldsymbol{v}.$$

We omit the details of the encryption algorithm, as this is just a thought experiment. The scheme is not secure against any adversary who has taken Linear Algebra 101.

That is, the message μ is the eigenvalue of the ciphertext $\boldsymbol{C}$ corresponding to the (secret) eigenvector $\boldsymbol{v}$. To verify that this is indeed a ring homomorphism, consider two ciphertexts $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ such that $\boldsymbol{C}_1\boldsymbol{v} = \mu_1\boldsymbol{v}$ and $\boldsymbol{C}_2\boldsymbol{v} = \mu_2\boldsymbol{v}$. We define the ring operations on ciphertexts naturally by setting $\boldsymbol{C}_1 \boxplus \boldsymbol{C}_2 := \boldsymbol{C}_1 + \boldsymbol{C}_2$ and $\boldsymbol{C}_1 \boxtimes \boldsymbol{C}_2 := \boldsymbol{C}_1\boldsymbol{C}_2$. Then, the scheme indeed forms a ring homomorphism, as shown below.

These are simply matrix addition and matrix multiplication.

$$\begin{aligned}(\boldsymbol{C}_1 + \boldsymbol{C}_2)\boldsymbol{v} &= \boldsymbol{C}_1\boldsymbol{v} + \boldsymbol{C}_2\boldsymbol{v} = \mu_1\boldsymbol{v} + \mu_2\boldsymbol{v} = (\mu_1 + \mu_2)\boldsymbol{v} \\ (\boldsymbol{C}_1\boldsymbol{C}_2)\boldsymbol{v} &= \boldsymbol{C}_1(\boldsymbol{C}_2\boldsymbol{v}) = \boldsymbol{C}_1(\mu_2\boldsymbol{v}) = \mu_2(\boldsymbol{C}_1\boldsymbol{v}) = (\mu_1\mu_2)\boldsymbol{v}\end{aligned}$$

Noise to the Rescue

Of course, the previous scheme is not secure against any adversary with a basic understanding of linear algebra. We address this by modifying it into a *noisy* version, while carefully ensuring that it remains *somewhat* homomorphic. That is, we consider a scheme in which a ciphertext encrypting a message $\mu \in \mathbb{Z}_q$ under a secret key

The tension between structure and hardness, and between functionality and privacy.

Someone might find this abrupt—why suddenly consider a noisy variant? However, this is a well-established approach in constructing FHE. Recall also the LPN problem.

$\boldsymbol{v} \in \mathbb{Z}_q^m$ is a matrix $\boldsymbol{C} \in \mathbb{Z}_q^{m \times m}$ satisfying the following condition for some *small* noise $\boldsymbol{e} \in \mathbb{Z}_q^m$.

$$\boldsymbol{C}\boldsymbol{v} = \mu\boldsymbol{v} + \boldsymbol{e}$$

Here, *small* means that it has a small norm if we take $\{-\lceil q/2 \rceil + 1, \ldots, \lfloor q/2 \rfloor\}$ as the representatives of $\mathbb{Z}_q$.

This noisy variant raises several questions that need to be addressed.

1. Is it still (somewhat) homomorphic?
2. How can we decrypt in the presence of noise?
3. How can we encrypt?
4. Is this scheme secure?

We will answer each of these questions in turn.

Homomorphism. Let's begin by examining whether this scheme is (somewhat) homomorphic even in the presence of noise. Let $\boldsymbol{C}_1\boldsymbol{v} = \mu_1\boldsymbol{v} + \boldsymbol{e}_1$ and $\boldsymbol{C}_2\boldsymbol{v} = \mu_2\boldsymbol{v} + \boldsymbol{e}_2$. We define the ring operations on ciphertexts naturally, as before. First, let's check addition. The following equations show that $\boldsymbol{C}_1 \boxplus \boldsymbol{C}_2$ is a ciphertext encrypting $\mu_1 + \mu_2$, but with noise $\boldsymbol{e}_1 + \boldsymbol{e}_2$.

$$\begin{aligned}(\boldsymbol{C}_1 + \boldsymbol{C}_2)\boldsymbol{v} &= \boldsymbol{C}_1\boldsymbol{v} + \boldsymbol{C}_2\boldsymbol{v} \\ &= (\mu_1\boldsymbol{v} + \boldsymbol{e}_1) + (\mu_2\boldsymbol{v} + \boldsymbol{e}_2) \\ &= (\mu_1 + \mu_2)\boldsymbol{v} + (\boldsymbol{e}_1 + \boldsymbol{e}_2)\end{aligned}$$

Is $\boldsymbol{e}_1 + \boldsymbol{e}_2$ small when $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are small? Yes, if $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are *sufficiently* small, then $\boldsymbol{e}_1 + \boldsymbol{e}_2$ should also remain small. By starting with ciphertexts that have *smaller* noise—something that can be controlled during the encryption procedure—the resulting ciphertext after addition will have small noise. Of course, if we keep adding ciphertexts, eventually the noise will grow too large, but this is acceptable since we are only aiming for *somewhat* homomorphism.

Next, multiplication. The following equations show that $\boldsymbol{C}_1 \boxtimes \boldsymbol{C}_2$ is a ciphertext encrypting $\mu_1\mu_2$, but with noise $\boldsymbol{e}_\times = \mu_2\boldsymbol{e}_1 + \boldsymbol{C}_1\boldsymbol{e}_2$.

Notice asymmetry here. Such asymmetry is exploited in certain results, such as:

Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, 2014

$$\begin{aligned}(\boldsymbol{C}_1\boldsymbol{C}_2)\boldsymbol{v} &= \boldsymbol{C}_1(\boldsymbol{C}_2\boldsymbol{v}) \\ &= \boldsymbol{C}_1(\mu_2\boldsymbol{v} + \boldsymbol{e}_2) \\ &= \mu_2(\boldsymbol{C}_1\boldsymbol{v}) + \boldsymbol{C}_1\boldsymbol{e}_2 \\ &= \mu_2(\mu_1\boldsymbol{v} + \boldsymbol{e}_1) + \boldsymbol{C}_1\boldsymbol{e}_2 \\ &= (\mu_1\mu_2)\boldsymbol{v} + (\mu_2\boldsymbol{e}_1 + \boldsymbol{C}_1\boldsymbol{e}_2)\end{aligned}$$

Is $\boldsymbol{e}_\times$ small when $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are small? Not necessarily. If the ciphertext $\boldsymbol{C}_1$ has large components, then $\boldsymbol{e}_\times$ could also be large. Similarly, if the message μ_2 is large, then $\boldsymbol{e}_\times$ could become large as well.

To address this, we need to restrict both the message space and the ciphertext space to consist only of small elements. The message space is not a major concern for us; since our blueprint only requires a homomorphic NAND gate, we can restrict the message space to $\{0,1\}$.

However, this can be a significant drawback from a practical perspective. Implementing every homomorphic circuit solely with NAND gates is cumbersome and leads to inefficiency.

Decryption. How can we decrypt in the presence of noise? This is now easy to answer because we restricted the message space to be $\{0,1\}$. The product of a ciphertext $\boldsymbol{C}$ and its corresponding secret eigenvector $\boldsymbol{v}$, given by $\boldsymbol{Cv} = \mu\boldsymbol{v} + \boldsymbol{e}$, would be $\boldsymbol{v} + \boldsymbol{e}$ if $\mu = 1$, and $\boldsymbol{e}$ if $\mu = 0$. Since $\boldsymbol{e}$ is a small noise vector, if $\boldsymbol{v}$ has at least one *large* component—which will be ensured by our design—then these two cases can be easily distinguished by checking whether $\boldsymbol{Cv}$ has at least one large component or not.

Learning with Errors

To address the remaining questions, we have to review some related concepts. First, we introduce the Learning with Errors (LWE) problem[6], on which the security of the GSW scheme is based. The problem $LWE_{n,q,\chi,m}$ is to distinguish between the following two distributions, defined with respect to a uniformly random secret $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$.

[6] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005

1. $(\boldsymbol{A}, \boldsymbol{b})$ where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$, $\boldsymbol{e} \leftarrow \chi^m$, and $\boldsymbol{b} \leftarrow \boldsymbol{As} + \boldsymbol{e}$
2. $(\boldsymbol{A}, \boldsymbol{b})$ where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$ and $\boldsymbol{b} \leftarrow \mathbb{Z}_q^m$

Note that $LWE_{n,q,\chi,m}$ is information-theoretically impossible to solve when χ is the uniform distribution over Z_q and extremely easy to solve when χ outputs constantly zero (by Gaussian elimination). We are interested in the intermediate case, where χ is a *small* noise distribution. In this note, you can think of χ as a uniform distribution over $[-B, B]$, where $B \ll q/2$.

Gadget Decomposition

We now introduce *gadget decomposition* and related notations, a useful tool for describing the GSW scheme and, more generally, lattice-based cryptosystems. At a high level, gadget decomposition allows us to transform a matrix with large components into a higher-dimensional matrix with small components, while ensuring that it can later be recomposed using only *linear* operations. In this note, you can think of it simply as a fancy term for *bit-decomposition*.

To be precise, fix a *gadget vector* $\boldsymbol{g} \in \mathbb{Z}_q^k$. A typical choice, which aligns with the bit-decomposition setting we will use in this note, is the following, with $k = \lceil \log q \rceil$.

$$\boldsymbol{g} = \begin{pmatrix} 1 \\ 2 \\ 2^2 \\ \vdots \\ 2^{k-1} \end{pmatrix}$$

We first define the gadget composition function g with respect to $\boldsymbol{g}$ as follows.

$$\begin{aligned} g : \mathbb{Z}_q^{n \times km} &\to \mathbb{Z}_q^{n \times m} \\ \boldsymbol{M} &\mapsto \boldsymbol{M}\boldsymbol{G}_m \end{aligned}$$

Here, $\boldsymbol{G}_m$ is the gadget matrix, defined as $\boldsymbol{I}_m \otimes \boldsymbol{g}$, where $\otimes$ denotes the tensor product (also known as the Kronecker product). We will omit the subscript when the context makes it clear. In our case, g is simply the bit-composition function.

https://en.wikipedia.org/wiki/Kronecker_product

We call $g^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}_q^{n \times km}$ a *gadget decomposition* algorithm with respect to $\boldsymbol{g}$ if the followings hold.

1. For any $\boldsymbol{M}$, we have
$$g \circ g^{-1}(\boldsymbol{M}) = \boldsymbol{M}.$$
2. The components of the outputs of g^{-1} are all *small*.

In this note, we will simply use bit-decomposition, which outputs a binary matrix that satisfies the above conditions.

Note that g^{-1} is only a right inverse of g and not a left inverse. That is, $g^{-1} \circ g(\boldsymbol{M}) \stackrel{?}{=} \boldsymbol{M}$ does *not* hold in general. However, the operation $g^{-1} \circ g$ is still interesting enough to warrant attention. If we dentoe $\boldsymbol{M}' := g^{-1} \circ g(\boldsymbol{M})$, then $g(\boldsymbol{M}') = g(\boldsymbol{M})$ holds, while components of $\boldsymbol{M}'$ are ensured to be small, as it is an output of g^{-1}. It seems reasonable to refer to this as the *gadget re-decomposition* algorithm.

This operation is commonly referred to as *flattening* in the literature.

Encryption & Security

With LWE and gadget decomposition in hand, we are now ready to discuss the encryption algorithm of the GSW scheme and its security. Recall that our goal is to, given a message $\mu \in \{0, 1\}$, output *small* matrix $\boldsymbol{C} \in \mathbb{Z}_p^{m \times m}$ that satisfies the following for some small noise vector $\boldsymbol{e} \in \mathbb{Z}_q^m$. Importantly, for security, the secret eigenvector $\boldsymbol{v}$ should not be leaked from the ciphertext $\boldsymbol{C}$.

$$\boldsymbol{C}\boldsymbol{v} = \mu \boldsymbol{v} + \boldsymbol{e}$$

Note that, in the case of $\mu = 0$, our goal reduces to producing a pseudorandom small matrix $\boldsymbol{C}$ satisfying $\boldsymbol{C}\boldsymbol{v} \approx \boldsymbol{0}$. This follows almost immediately from LWE and gadget decomposition. Consider an LWE sample with a uniformly random secret $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$: Sample a uniform random matrix $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$ and *small* noise $\boldsymbol{e} \in \mathbb{Z}_q^m$. Then, $\hat{\boldsymbol{A}} = (\boldsymbol{A}||\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ is pseudorandom over $\mathbb{Z}_q^{m\times(n+1)}$ under LWE assumption. Furthermore, if we define $\hat{\boldsymbol{s}} = (-\boldsymbol{s}, 1) \in \mathbb{Z}_q^{n+1}$, we have $\hat{\boldsymbol{A}}\hat{\boldsymbol{s}} = \boldsymbol{e} \approx \boldsymbol{0}$. Also, note the following equality from gadget decomposition.

$$\begin{aligned}\hat{\boldsymbol{A}}\hat{\boldsymbol{s}} &= g \circ g^{-1}(\hat{\boldsymbol{A}})\hat{\boldsymbol{s}} \\ &= (g^{-1}(\hat{\boldsymbol{A}})\boldsymbol{G})\hat{\boldsymbol{s}} \\ &= g^{-1}(\hat{\boldsymbol{A}})(\boldsymbol{G}\hat{\boldsymbol{s}})\end{aligned}$$

Now, if we set $m = (n+1)k$, then $\boldsymbol{C} = g^{-1}(\hat{\boldsymbol{A}}) \in \mathbb{Z}_q^{m\times m}$ is a pseudorandom small matrix, satisfying $\boldsymbol{C}\boldsymbol{v} \approx \boldsymbol{0}$ for $\boldsymbol{v} = \boldsymbol{G}\hat{\boldsymbol{s}} \in \mathbb{Z}_q^m$ as we desired.

For the case of $\mu = 1$, one might consider simply returning $g^{-1}(\hat{\boldsymbol{A}}) + \mu\boldsymbol{I}_m$. However, this naive approach may leave noticeable traces on the diagonal entries, since $g^{-1}(\hat{\boldsymbol{A}})$ is not pseudorandom over the entire $\mathbb{Z}_q^{m\times m}$, but only over a subset of small matrices. A straightforward fix is to apply re-decomposition, which can be further simplified as shown below.

$$\begin{aligned}g^{-1} \circ g(g^{-1}(\hat{\boldsymbol{A}}) + \mu\boldsymbol{I}_m) &= g^{-1}\left((g^{-1}(\hat{\boldsymbol{A}}) + \mu\boldsymbol{I}_m)\boldsymbol{G}\right) \\ &= g^{-1}\left(g^{-1}(\hat{\boldsymbol{A}})\boldsymbol{G} + \mu\boldsymbol{G}\right) \\ &= g^{-1}(\hat{\boldsymbol{A}} + \mu\boldsymbol{G})\end{aligned}$$

Putting Everything Together

Below is a summary of the scheme. First, set (n, q, χ) so that $\mathsf{LWE}_{n,q,\chi,m}$ is hard, where $m = (n+1)k$ for $k = \lceil \log q \rceil$. To support homomorphism, the error distribution χ must be small.

- $\mathsf{KeyGen}(1^\lambda)$: Sample uniform random $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and output it as the secret key $\mathsf{sk} = \boldsymbol{s}$.
- $\mathsf{Enc}_{\mathsf{sk}}(\mu)$: Given a message $\mu \in \{0, 1\}$ as input, sample a uniform random matrix $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$ and *small* noise $\boldsymbol{e} \leftarrow \chi^m$. Set $\hat{\boldsymbol{A}} = (\boldsymbol{A}||\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \in \mathbb{Z}_q^{m\times(n+1)}$. Output $\boldsymbol{C} = g^{-1}(\hat{\boldsymbol{A}} + \mu\boldsymbol{G}) \in \mathbb{Z}_q^{m\times m}$.
- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{C})$: Given a ciphertext $\boldsymbol{C} \in \mathbb{Z}_q^{m\times m}$ as input, compute $\boldsymbol{m} = \boldsymbol{C}(\boldsymbol{G}\hat{\boldsymbol{s}})$, where $\hat{\boldsymbol{s}} = (-\boldsymbol{s}, 1) \in \mathbb{Z}_q^{n+1}$. Output 0 if every component of $\boldsymbol{m}$ is *small*, and output 1 otherwise.
- $\boldsymbol{C}_1 \boxplus \boldsymbol{C}_2$: Output $\boldsymbol{C}_1 + \boldsymbol{C}_2$.

- $\boldsymbol{C}_1 \boxtimes \boldsymbol{C}_2$: Output $\boldsymbol{C}_1\boldsymbol{C}_2$.

Better Noise Control

While the above scheme already provides non-trivial homomorphism, it is not sufficient to be bootstrappable. While we are not going to do precise noise analysis, the main issue is that the components of the ciphertexts grow *double-exponentially* with respect to the (multiplicative) depth of the circuit. We can address this issue by applying *gadget re-decomposition* after each arithmetic gate, ensuring that the ciphertext components remain small. This yields the bootstrappable GSW SHE scheme.

Larger parameters require greater depth, which in turn requires even larger parameters, and so on.

- $\boldsymbol{C}_1 \boxplus \boldsymbol{C}_2$: Output $g^{-1} \circ g\,(\boldsymbol{C}_1 + \boldsymbol{C}_2)$.
- $\boldsymbol{C}_1 \boxtimes \boldsymbol{C}_2$: Output $g^{-1} \circ g\,(\boldsymbol{C}_1\boldsymbol{C}_2)$.

Simplified Version

In fact, we can simplify the above description of the GSW scheme as follows, which is the version most research papers adopt. While this provides a more compact presentation, I personally find the earlier version more accessible at first glance, particularly for understanding the structure of the homomorphism through eigenvalues. Therefore, we chose to present the earlier version first. We leave it as an exercise to verify that the following scheme is essentially equivalent to the previous description of the GSW scheme.

More precisely, the PKE version of it.

For instance, a ciphertext of the scheme below is not a square matrix.

- $\mathsf{KeyGen}(1^\lambda)$: Sample uniform random $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and output it as the secret key $\mathsf{sk} = \boldsymbol{s}$.
- $\mathsf{Enc}_{\mathsf{sk}}(\mu)$: Given a message $\mu \in \{0,1\}$ as input, sample a uniform random matrix $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$ and *small* noise $\boldsymbol{e} \leftarrow \chi^m$. Set $\hat{\boldsymbol{A}} = (\boldsymbol{A}||\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \in \mathbb{Z}_q^{m\times(n+1)}$. Output $\boldsymbol{C} = \hat{\boldsymbol{A}} + \mu\boldsymbol{G}$.
- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{C})$: Given a ciphertext $\boldsymbol{C} \in \mathbb{Z}_q^{m\times(n+1)}$ as input, compute $\boldsymbol{m} = \boldsymbol{C}\hat{\boldsymbol{s}}$, where $\hat{\boldsymbol{s}} = (-\boldsymbol{s}, 1) \in \mathbb{Z}_q^{n+1}$. Output 0 if every component of $\boldsymbol{m}$ is *small*, and output 1 otherwise.
- $\boldsymbol{C}_1 \boxplus \boldsymbol{C}_2$: Output $\boldsymbol{C}_1 + \boldsymbol{C}_2$.
- $\boldsymbol{C}_1 \boxtimes \boldsymbol{C}_2$: Output $g^{-1}\,(\boldsymbol{C}_1)\,\boldsymbol{C}_2$.

References

Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, 2014.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, 2010.