SCIS 2015 The 32nd Symposium on
Cryptography and Information Security
Kokura, Japan, Jan. 20 - 23, 2015
The Institute of Electronics,
Information and Communication Engineers

SIMD Operations in GSW-FHE

Ryo Hiromasa [*] Masayuki Abe [†*] Tatsuaki Okamoto [†*]

Abstract: We construct the first single-instruction-multiple-data (SIMD) variant of GSW-FHE that is a fully homomorphic encryption (FHE) scheme provided by Gentry, Sahai, and Waters (CRYPTO 2013). Our scheme can encrypt a message *vector* into one ciphertext and homomorphically compute SIMD operations. All of the previous SIMD FHE schemes have complex and expensive homomorphic multiplication algorithms involving the dimension or modulus reduction. In our scheme, homomorphic SIMD addition and multiplication are just matrix addition and multiplication.

Keywords: Fully Homomorphic Encryption, Lattice-based Cryptography, Learning with Errors

1 Introduction

Fully homomorphic encryption (FHE) allows us to evaluate any function over encrypted data by only using public information. This can be used, for example, to outsource computations to remote servers without compromising privacy. Since the breakthrough work by Gentry [Gen09a, Gen09b], many different varieties of FHE have been proposed [DGHV10, BV11a, BV11b, BGV12, Bra12, LTV12, GSW13]. To date, the fastest (and simplest) FHE based on the *standard* lattice assumption is the one by Gentry et al. [GSW13] (in the following, we call this scheme GSW-FHE). However, it is required to take a heavy cost for evaluating a large number of ciphertexts. The way to deal with this issue is to *pack* multiple messages into one ciphertext.

Packing messages allows us to apply *single-instruction-multiple-data (SIMD)* homomorphic operations to all encrypted messages. In the case where a remote server stores encrypted data and we want to retrieve certain data from this server, we first apply an equality function to every encrypted data. If the stored data were packed into one ciphertext, we can do that by only one homomorphic evaluation of the equality function. Smart and Vercautren [SV10], for the first time, showed that the message space of the ideal-lattice-based FHE [Gen09a, Gen09b] (and polynomial-ring-based FHE [BV11b, BGV12]) can be partitioned into a vector of *plaintext slots* by applying the Kummer-Dedekind theorem to the ideal generated by the plaintext moduli. On standard-lattice-based FHE, Brakerski et al. [BGH13] packed messages in the FHE variant [BV11a,BGV12,Bra12] of Regev's encryption [Reg05], using the result in [PVW08] that described the way to pack in [Reg05]. Cheon et al. [CCK$^+$13] observed that the integer-based FHE [DGHV10] can also compute SIMD homomorphic operations.

1.1 Our Results

In this paper, we construct the first SIMD fully homomorphic encryption scheme based on [GSW13] that is secure under the standard lattice assumption. Our SIMD FHE can encrypt a message *vector* into one ciphertext and homomorphically compute SIMD operations. All of the previous SIMD FHE schemes have complex and expensive homomorphic multiplication algorithms involving the dimension or modulus reduction. In our scheme, homomorphic SIMD addition and multiplication are just matrix addition and multiplication.

1.2 Our Techniques

Peikert et al. [PVW08] showed that Regev's encryption [Reg05] based on the standard lattice assumption can be converted into a scheme that can encrypt multiple messages in one ciphertext. Using this technique, [BGH13] constructed the standard-LWE based FHE [BV11a, BGV12] that can also encrypt multiple messages and evaluate SIMD homomorphic operations. As described in [Bar10, Rot11], symmetric re-randomizable homomorphic encryption can simply be converted into an asymmetric one, and [BV14] introduced that [GSW13] has the ability to re-randomize its ciphertexts. Our translation first constructs a *symmetric* SIMD variant of [GSW13] along the lines of [PVW08, BGH13]. To transform it to an asymmetric one, we publish the symmetric encryption of all message vectors as the public key, and then generate the ciphertext by re-randomizing the symmetric encryption corresponding to the message vector. Obviously, our construction has the significantly large public key (whose size is exponential in the number of encrypted bits r!), so we can only set $r = O(\log \lambda)$.

Our transformation allows us to proceed SIMD homomorphic addition and multiplication in [GSW13], but is insufficient to bring out the full potential of our scheme. We now consider the following three gates r-Add, r-Mult, and r-$\mathsf{Permute}$: r-Add takes as input two ciphertexts that hold messages $(m_1, \ldots, m_r)$ and $(m'_1, \ldots, m'_r)$, respectively, and outputs a new ciphertext with the message $(m_1+m'_1, \ldots, m_r+m'_r)$; r-Mult takes as input the same messages as the above and outputs a new ciphertext with the message $(m_1 \cdot m'_1, \ldots, m_r \cdot m'_r)$; r-$\mathsf{Permute}$ transforms the ciphertext with $(m_1, \ldots, m_r)$

[*] Kyoto University

[†] NTT Secure Platform Laboratories

into the one with $(m_{\pi(1)},\dots,m_{\pi(r)})$ for a permutation π. Gentry et al. [GHS12] showed that the set of gates r-Add, r-Mult, and r-Permute allows us to efficiently evaluate complex circuits.

r-Add and r-Mult have already been implemented in our SIMD scheme, so far. Constructing the plaintext-slot switching procedure to implement r-Permute is slightly non-trivial. Let $\boldsymbol{C}$ be a packed ciphertext that encrypts the message vector $(m_1,\dots,m_r)$ under the secret key matrix $\boldsymbol{S}$, and π be a permutation. Every row of $\boldsymbol{S}$, which we call $\boldsymbol{s}_i^T$ for $i = 1,\dots,r$, corresponds to one of the encrypted messages m_i, and satisfies $\boldsymbol{s}_i^T\boldsymbol{C} = \text{noise} + m_i \cdot \boldsymbol{s}_i^T\boldsymbol{G}$ for some fixed matrix $\boldsymbol{G}$. Note that in this paper we consider permutations of row vectors in the matrix, that is, $\pi(\boldsymbol{S}) = [\boldsymbol{s}_{\pi(1)} \parallel \cdots \parallel \boldsymbol{s}_{\pi(r)}]^T$. Let us consider the decryption of $\boldsymbol{C}$ using the *permuted* secret key $\pi(\boldsymbol{S})$:

$$\pi(\boldsymbol{S})\boldsymbol{C} = \text{noise} + \left(\begin{array}{c} m_{\pi(1)} \cdot \boldsymbol{s}_{\pi(1)}^T \\ \hline \vdots \\ \hline m_{\pi(r)} \cdot \boldsymbol{s}_{\pi(r)}^T \end{array}\right)\boldsymbol{G}.$$

Since the decryption using $\pi(\boldsymbol{S})$ disorders the messages and secret keys, it is insufficient to publish a key switching gadget from $\pi(\boldsymbol{S})$ to $\boldsymbol{S}$ similar to [BGH13]. To complete the plaintext slot permutation procedure, we need to reorder only the secret keys by π^{-1}. Now we have seen from the above that multiplying $\pi(\boldsymbol{S})$ to the ciphertext under $\boldsymbol{S}$ leads to the disordering of the secret keys by π. This tells us that multiplying $\pi(\boldsymbol{S})$ to the ciphertext under $\pi^{-1}(\boldsymbol{S})$ yields a decryption under $\boldsymbol{S}$. To reorder only the secret keys (i.e., without reordering the messages), we publish something like a ciphertext with the messages $(1,\dots,1)$ under $\pi^{-1}(\boldsymbol{S})$ (that is, a key switching gadget from $\pi^{-1}(\boldsymbol{S})$ to $\boldsymbol{S}$) and homomorphically multiply it to $\boldsymbol{C}$ from the right. This fixes the order of the secret key vectors as $\pi(\pi^{-1}(\boldsymbol{S}))$ and multiplying the messages $(1,\dots,1)$ does not affect anything in the encrypted message vector $(m_{\pi(1)},\dots,m_{\pi(r)})$. One nice feature of our plaintext-slot switching is not to suffer from the inconvenience of security as in [BGH13]: we do not have to use a larger modulus than the underlying encryption scheme.

2 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, and the set of real numbers by $\mathbb{R}$. Let $\mathbb{G}$ be a group and $\mathcal{P}$ be a probability distribution, then we use $a \overset{U}{\leftarrow} \mathbb{G}$ to denote that a is chosen from $\mathbb{G}$ uniformly at random, and use $b \overset{R}{\leftarrow} \mathcal{P}$ to denote that b is chosen along $\mathcal{P}$. We take all logarithms to base 2, unless otherwise noted.

We assume that vectors are in column form and are written by bold lower-case letters, e.g., $\boldsymbol{x}$, and the i-th element of a vector is denoted by x_i. We denote the ℓ_∞ norm (the maximum norm) of the vector $\boldsymbol{x}$ by $\|\boldsymbol{x}\|_\infty$, and the ℓ_2 norm (the Euclidean norm) of $\boldsymbol{x}$ by $\|\boldsymbol{x}\|_2$. The inner product between two vectors is denoted by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Matrices are written by bold capital letters, e.g., $\boldsymbol{X}$, and the i-th column vector of a matrix is denoted by $\boldsymbol{x}_i$. For a matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, the notation $\boldsymbol{X}^T \in \mathbb{R}^{n\times m}$ denotes the transpose of $\boldsymbol{X}$. For two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n_1}$ and $\boldsymbol{B} \in \mathbb{R}^{m\times n_2}$, $[\boldsymbol{A} \parallel \boldsymbol{B}] \in \mathbb{R}^{m\times(n_1+n_2)}$ denotes the concatenation of $\boldsymbol{A}$ with $\boldsymbol{B}$. When we refer to the $n \times n$ identity matrix, we denote it by $\boldsymbol{I}_n$. For any vector $\boldsymbol{x} \in \mathbb{Z}^n$, we use $\boldsymbol{y} = [\boldsymbol{x}]_q$ to represent the vector for which $y_i = x_i \bmod q$ for every $i \in [n]$.

2.1 Learning with Errors

The *learning with errors (LWE) assumption* was first introduced by Regev [Reg05].

Definition 2.1 (DLWE). *For security parameter λ, let $n := n(\lambda)$ be an integer dimension, let $q := q(\lambda) \geq 2$ be an integer modulus, and let $\chi := \chi(\lambda)$ be an error distribution over $\mathbb{Z}$. $\mathsf{DLWE}_{n,q,\chi}$ is the problem to distinguish the following two distributions: In the first distribution, a tuple $(\boldsymbol{a}_i, b_i)$ is sampled from uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_q$; In the second distribution, $\boldsymbol{s} \overset{U}{\leftarrow} \mathbb{Z}_q^n$, and then a tuple $(\boldsymbol{a}_i, b_i)$ is sampled by sampling $\boldsymbol{a}_i \overset{U}{\leftarrow} \mathbb{Z}_q^n$, $e_i \overset{R}{\leftarrow} \chi$, and setting $b_i := \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod q$. The $\mathsf{DLWE}_{n,q,\chi}$ assumption is that $\mathsf{DLWE}_{n,q,\chi}$ is infeasible.*

Recall that GapSVP_γ is the promise problem to distinguish between the case in which the lattice has a vector shorter than $r \in \mathbb{Q}$, and the case in which all the lattice vectors are greater that $\gamma \cdot r$. SIVP_γ is the problem to find the set of short linearly independent vectors in a lattice. $\mathsf{DLWE}_{n,q,\chi}$ has reductions to the standard lattice assumptions as follows. These reductions take χ to be a discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$ (that is centered around 0 and has parameter αq for some $\alpha < 1$), which is statistically indistinguishable from a B-bounded distribution (i.e., $\mathbb{E}[X] = 0$ and $|X| \leq B$) for an appropriate B.

Corollary 2.1 (stated as Corollary 2.6 from [BV14] or Corollary 2.1 from [Bra12]). *Let $q := q(n) \in \mathbb{N}$ be a powers of primes $q := p^r$ or a product of distinct prime numbers $q := \prod_i q_i$ ($q_i := \mathrm{poly}(n)$ for all i), and let $\alpha \geq \sqrt{n}/q$. If there exists an efficient algorithm that solves (average-case) $\mathsf{DLWE}_{n,q,D_{\mathbb{Z},\alpha q}}$,*

- *there exists an efficient quantum algorithm that can solve $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$ and $\mathsf{SIVP}_{\tilde{O}(n/\alpha)}$ in the worst-case for any n-dimensional lattices.*
- *if in addition we have $q \geq \tilde{O}(2^{n/2})$, there exists an efficient classical algorithm that can solve $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$ in the worst-case for any n-dimensional lattices.*

2.2 Subgaussian

A real random variable X is subgaussian with parameter s if for all $t \in \mathbb{R}$, its (scaled) moment generating function holds $\mathbb{E}[\exp(2\pi t X)] \leq \exp(\pi s^2 t^2)$. Any B-bounded (centered) random variable X is subgaussian with parameter $B \cdot \sqrt{2\pi}$. Subgaussian random variables have the following two properties that can be easily obtained from the definition of subgaussian random variables:

- Homogeneity: If the subgaussian random variable X has parameter s, then cX is subgaussian with parameter cs.
- Pythagorean additivity: For two subgaussian random variables X_1 and X_2 (that is independent from X_1) with parameter s_1 and s_2, respectively, $X_1 + X_2$ is subgaussian with parameter $\sqrt{s_1^2 + s_2^2}$.

The above can be extended to vectors. A real random vector $\boldsymbol{x}$ is subgaussian with parameter s if for all real unit vectors $\boldsymbol{u}$, their marginal $\langle \boldsymbol{u}, \boldsymbol{x} \rangle$ is subgaussian with parameter s. It is clear from the definition that the concatenation of subgaussian variables or vectors, each of which has parameter s and is independent of the prior one, is also subgaussian with parameter s. The homogeneity and Pythagorean additivity also holds from the linearity of vectors. It is known that the euclidean norm of the subgaussian random vector has the following upper bound.

Lemma 2.1 ([Ver12]). *Let $\boldsymbol{x} \in \mathbb{R}^n$ be a random vector that has independent subgaussian coordinates with parameter s. Then there exists a universal constant C such that* $\Pr[\|\boldsymbol{x}\|_2 > C \cdot s\sqrt{n}] \le 2^{-\Omega(n)}$.

To suppress the noise growth, Gentry et al. [GSW13] made use of the procedure that decomposes a vector in binary representation. Alperin-Sheriff and Peikert [AP14] observed that instead of the decomposition procedure, using the following algorithm $\boldsymbol{G}^{-1}$ that samples a subgaussian random vector allows us to re-randomize errors in ciphertexts and tightly analyse the noise growth in [GSW13]. Let $\boldsymbol{g}^T := (1, 2, 2^2, \ldots, 2^{\lceil \log q \rceil - 1})$ and $\boldsymbol{G} := \boldsymbol{g}^T \otimes \boldsymbol{I}_n$.

Lemma 2.2 ([AP14], which is adapted from [MP12]). *There is a randomized, efficiently computable function $\boldsymbol{G}^{-1} : \mathbb{Z}_q^n \to \mathbb{Z}^{n \cdot \lceil \log q \rceil}$ such that for any $\boldsymbol{a} \in \mathbb{Z}_q^n$, $\boldsymbol{x} \overset{R}{\leftarrow} \boldsymbol{G}^{-1}(\boldsymbol{a})$ is subgaussian with parameter $O(1)$ and $\boldsymbol{a} = [\boldsymbol{G}\boldsymbol{x}]_q$*

3 SIMD GSW-FHE

We translate [GSW13] to be able to evaluate SIMD homomorphic operations. In Section 3.1, we present the SIMD encryption scheme. How to permute its plaintext slots is discussed in Section 3.2.

Our translation first constructs the symmetric SIMD scheme of [GSW13] and then transforms it to an asymmetric one by the method described in [Bar10, Rot11]. In addition, we describe how to permute a packed plaintext vector in our scheme.

3.1 Construction

Let λ be the security parameter. Our SIMD scheme is parametrized by an integer lattice dimension n, integer modulus q, and a distribution χ over $\mathbb{Z}$ that is assumed to be subgaussian , all of which depend on λ. We let $\ell := \lceil \log q \rceil$, $m := O(n \log q)$ [1] , and $N := (n + r) \cdot \ell$. Let r be the number of bits to be encrypted, which defines the message space $\{0, 1\}^r$. The ciphertext space is $\mathbb{Z}_q^{(n+r)\times N}$. Our scheme uses the rounding function $\lfloor \cdot \rceil_2$ that for any $x \in \mathbb{Z}_q$, $\lfloor x \rceil_2$ outputs 1 if x is close to $q/4$, and 0 otherwise. Recall that $\boldsymbol{g}^T = (1, 2, \ldots, 2^{\ell-1})$ and $\boldsymbol{G} = \boldsymbol{g}^T \otimes \boldsymbol{I}_n$.

- $\mathsf{KeyGen}(1^\lambda, r)$: Set parameters n, q, m, ℓ, N, and χ as described above. Sample a uniformly random matrix $\boldsymbol{A} \overset{U}{\leftarrow} \mathbb{Z}_q^{n\times m}$, secret key matrix $\boldsymbol{S}' \overset{R}{\leftarrow} \chi^{r\times n}$, and noise matrix $\boldsymbol{E} \overset{R}{\leftarrow} \chi^{r\times m}$. Let $\boldsymbol{S} := [\boldsymbol{I}_r \parallel -\boldsymbol{S}'] \in \mathbb{Z}_q^{r\times(n+r)}$. We denote by $\boldsymbol{s}_i^T$ the i-th row of $\boldsymbol{S}$. Set
$$\boldsymbol{B} := \left[\left(\begin{array}{c} \boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E} \\ \hline \boldsymbol{A} \end{array}\right)\right]_q \in \mathbb{Z}_q^{(n+r)\times m}.$$
For all $\boldsymbol{m} \in \{0, 1\}^r$, first choose $\boldsymbol{R_m} \overset{U}{\leftarrow} \{0, 1\}^{m\times N}$ and set
$$\boldsymbol{P_m} := \left[\boldsymbol{B}\boldsymbol{R_m} + \left(\begin{array}{c} m_1 \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_r \cdot \boldsymbol{s}_r^T \\ \hline \boldsymbol{0} \end{array}\right)\boldsymbol{G}\right]_q \in \mathbb{Z}_q^{(n+r)\times N}.$$
Output $\mathsf{pk} := (\{\boldsymbol{P_m}\}_{\boldsymbol{m}\in\{0,1\}^r}, \boldsymbol{B})$ and $\mathsf{sk} := \boldsymbol{S}$.
- $\mathsf{SecEnc}_{\mathsf{sk}}(\boldsymbol{m})$: Sample random matrices $\boldsymbol{A}' \overset{U}{\leftarrow} \mathbb{Z}_q^{n\times N}$ and $\boldsymbol{E} \overset{R}{\leftarrow} \chi^{r\times N}$, parse $\boldsymbol{S} = [\boldsymbol{I}_r \parallel -\boldsymbol{S}']$, and output the ciphertext
$$\boldsymbol{C} := \left[\left(\begin{array}{c} \boldsymbol{S}'\boldsymbol{A}' + \boldsymbol{E} \\ \hline \boldsymbol{A}' \end{array}\right) + \left(\begin{array}{c} m_1 \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_r \cdot \boldsymbol{s}_r^T \\ \hline \boldsymbol{0} \end{array}\right)\boldsymbol{G}\right]_q \in \mathbb{Z}_q^{(n+r)\times N}.$$
- $\mathsf{PubEnc}_{\mathsf{pk}}(\boldsymbol{m})$: Sample a random matrix $\boldsymbol{R} \overset{U}{\leftarrow} \{0, 1\}^{m\times N}$, and output the ciphertext
$$\boldsymbol{C} := [\boldsymbol{B}\boldsymbol{R} + \boldsymbol{P_m}]_q \in \mathbb{Z}_q^{(n+r)\times N}.$$
- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{C})$: For $i = 1, \ldots, r$, let $\boldsymbol{c}_{i\ell-1}$ be the $i\ell - 1$-th column of $\boldsymbol{C}$, and compute $m_i' := \lfloor [\langle \boldsymbol{s}_i^T, \boldsymbol{c}_{i\ell-1} \rangle]_q \rceil_2 \in \{0, 1\}$. Output $(m_1', \ldots, m_r') \in \{0, 1\}^r$.
- $\boldsymbol{C}_1 \oplus \boldsymbol{C}_2$: Output $\boldsymbol{C}_{add} := [\boldsymbol{C}_1 + \boldsymbol{C}_2]_q \in \mathbb{Z}_q^{(n+r)\times N}$ as the result of homomorphic addition between the input ciphertexts.
- $\boldsymbol{C}_1 \odot \boldsymbol{C}_2$: Output $\boldsymbol{C}_{mult} := [\boldsymbol{C}_1 \boldsymbol{G}^{-1}(\boldsymbol{C}_2)]_q \in \mathbb{Z}_q^{(n+r)\times N}$ as the result of homomorphic multiplication between the input ciphertexts.

In the following, we state the correctness of our SIMD asymmetric encryption. The symmetric version can also be proven as well.

Proposition 3.1. *For every* $(\mathsf{pk}, \mathsf{sk}) \overset{R}{\leftarrow} \mathsf{KeyGen}(1^\lambda, r)$*, $\boldsymbol{m} \in \{0, 1\}^r$, and $\boldsymbol{C} \overset{R}{\leftarrow} \mathsf{PubEnc}_{\mathsf{pk}}(\boldsymbol{m})$ such that for all $i \in [r]$, $\|\boldsymbol{s}_i^T\boldsymbol{C} - m_i \cdot \boldsymbol{s}_i^T\boldsymbol{G}\|_\infty < q/8$, we have $\boldsymbol{m} = \mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{C})$.*

Proof. For $i = 1, \ldots, r$, it holds that
$$\begin{aligned} \boldsymbol{s}_i^T\boldsymbol{C} &= \boldsymbol{s}_i^T\left(\boldsymbol{B}\boldsymbol{R} + \boldsymbol{B}\boldsymbol{R_m} + \left(\begin{array}{c} m_1 \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_r \cdot \boldsymbol{s}_r^T \\ \hline \boldsymbol{0} \end{array}\right)\boldsymbol{G}\right) \\ &= \boldsymbol{e}_i^T(\boldsymbol{R} + \boldsymbol{R_m}) + m_i \cdot \boldsymbol{s}_i^T\boldsymbol{G}. \end{aligned}$$

Because of $\|\boldsymbol{e}_i^T(\boldsymbol{R} + \boldsymbol{R_m})\|_\infty < q/8$ and $2^{\ell-2} \in [q/4, q/2)$, we have $[\langle \boldsymbol{s}_i^T, \boldsymbol{c}_{i\ell-1} \rangle]_q \approx q/4$ if $m_i = 1$, $[\langle \boldsymbol{s}_i^T, \boldsymbol{c}_{i\ell-1} \rangle]_q \approx 0$ otherwise. □

[1] Rigorously, we must chose $m := O((n + r)\log q)$ from the leftover hash lemma. In practical parameter settings, we will set $n := \Omega(\lambda)$ and $r := O(\log \lambda)$. This is why we chose $m := O(n \log q)$.

The following two propositions mentioning the security of our scheme can be immediately proven from $\mathsf{DLWE}_{n,q,\chi}$ (and the circular security assumption).

Proposition 3.2. *Let $\boldsymbol{S}' \in \mathbb{Z}^{r\times n}$ be a matrix generated in* KeyGen*, and $\boldsymbol{A} \in \mathbb{Z}^{n\times m}, \boldsymbol{E} \in \mathbb{Z}^{r\times m}$ be a matrix generated in* SecEnc*. Then, the distribution $[(\boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E})^T \parallel \boldsymbol{A}^T]^T$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^{(n+r)\times N}$.*

Proposition 3.3. *Let $\boldsymbol{B} \in \mathbb{Z}^{(n+r)\times m}, \boldsymbol{R_m} \in \{0,1\}^{m\times N}$ be matrices generated in* KeyGen *and $\boldsymbol{R} \in \{0,1\}^{m\times N}$ be a matrix generated in* PubEnc*. Then, the joint probability distribution $(\boldsymbol{B}, \boldsymbol{BR_m}, \boldsymbol{BR})$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^{(n+r)\times m} \times \mathbb{Z}_q^{(n+r)\times N} \times \mathbb{Z}_q^{(n+r)\times N}$.*

We need to estimate the noise growth by the evaluation of SIMD homomorphic addition and multiplication. Similar to [AP14], we employ the properties of subgaussian random variables for a tight analysis. We collect the result of the estimation in the following proposition.

Proposition 3.4. *Let $\boldsymbol{S} \in \mathbb{Z}^{r\times(n+r)}$ be a secret key matrix. Let $\boldsymbol{C}_1 \in \mathbb{Z}_q^{(n+r)\times N}$ and $\boldsymbol{C}_2 \in \mathbb{Z}_q^{(n+r)\times N}$ be ciphertexts that encrypt message vectors $(m_{1,1}, \ldots, m_{1,r}) \in \{0,1\}^r$ and $(m_{2,1}, \ldots, m_{2,r}) \in \{0,1\}^r$ with noise matrices $\boldsymbol{E}_1 \in \mathbb{Z}^{r\times N}$ and $\boldsymbol{E}_2 \in \mathbb{Z}^{r\times N}$, respectively. Let $\boldsymbol{e}_{i,j}^T \in \mathbb{Z}^{1\times N}$ be the j-th row vector of $\boldsymbol{E}_i$ (for $i = 1, 2$ and $j = 1, \ldots, r$). Let $\boldsymbol{C}_{add} := \boldsymbol{C}_1 \oplus \boldsymbol{C}_2$ and $\boldsymbol{C}_{mult} \overset{R}{\leftarrow} \boldsymbol{C}_1 \odot \boldsymbol{C}_2$. Then, we have*

$$\boldsymbol{SC}_{add} = \boldsymbol{E}_{add} + \left(\begin{array}{c} (m_{1,1} + m_{2,1}) \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline (m_{1,r} + m_{2,r}) \cdot \boldsymbol{s}_r^T \end{array}\right) \boldsymbol{G} \in \mathbb{Z}_q^{r\times N},$$

$$\boldsymbol{SC}_{mult} = \boldsymbol{E}_{mult} + \left(\begin{array}{c} (m_{1,1} \cdot m_{2,1}) \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline (m_{1,r} \cdot m_{2,r}) \cdot \boldsymbol{s}_r^T \end{array}\right) \boldsymbol{G} \in \mathbb{Z}_q^{r\times N},$$

where $\boldsymbol{E}_{add} := \boldsymbol{E}_1 + \boldsymbol{E}_2$ and $\boldsymbol{E}_{mult}$ is of the form $\boldsymbol{E} + [m_{1,1} \cdot \boldsymbol{e}_{2,1} \parallel \cdots \parallel m_{1,r} \cdot \boldsymbol{e}_{2,r}]^T$. In particular, $\boldsymbol{E}$ has in the i-th row independent subgaussian entries with parameter $O(\|\boldsymbol{e}_{1,i}\|_2)$.

Proof. We can easily see that the statement for $\boldsymbol{C}_{add}$ holds. For $\boldsymbol{C}_{mult}$, we have

$$\begin{aligned} \boldsymbol{SC}_{mult} &= \boldsymbol{SC}_1 \boldsymbol{G}^{-1}(\boldsymbol{C}_2) \\ &= \left\{ \boldsymbol{E}_1 + \left(\begin{array}{c} m_{1,1} \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_{1,r} \cdot \boldsymbol{s}_r^T \end{array}\right) \boldsymbol{G} \right\} \boldsymbol{G}^{-1}(\boldsymbol{C}_2) \\ &= \boldsymbol{E}_1 \boldsymbol{G}^{-1}(\boldsymbol{C}_2) + \left(\begin{array}{c} m_{1,1} \cdot \boldsymbol{e}_{2,1}^T \\ \hline \vdots \\ \hline m_{1,r} \cdot \boldsymbol{e}_{2,r}^T \end{array}\right) + \left(\begin{array}{c} m_{1,1} m_{2,1} \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_{1,r} m_{2,r} \cdot \boldsymbol{s}_r^T \end{array}\right) \boldsymbol{G}. \end{aligned}$$

If we let $\boldsymbol{E} := \boldsymbol{E}_1 \boldsymbol{G}^{-1}(\boldsymbol{C}_2)$, the i-th row entries of $\boldsymbol{E}$ has parameter $O(\|\boldsymbol{e}_{1,i}\|_2)$ from the subgaussian properties and Lemma 2.2. □

Similar to the original GSW scheme, our SIMD scheme also has the asymmetric noise growth property, and thereby computing a polynomial length chain of homomorphic multiplications incurs the noise growth by a multiplicative polynomial factor. We set the following corollary immediately proven from Proposition 3.4 and the properties of subgaussian random variables. In the following corollary, we include the fixed ciphertext $\boldsymbol{G} \in \mathbb{Z}^{(n+r)\times N}$ of message vector $(1, \ldots, 1)$ with noise zero. This makes noise in the output ciphertext subgaussian and independent of noise in the input ciphertexts.

Corollary 3.1. *For $i = 1, \ldots, k$, let $\boldsymbol{C}_i \in \boldsymbol{Z}^{(n+r)\times N}$ be a packed ciphertext with noise matrix $\boldsymbol{E}_i \in \mathbb{Z}^{r\times N}$. Let*

$$\boldsymbol{C} \overset{R}{\leftarrow} \bigodot_{i=1}^{k} \boldsymbol{C}_i \odot \boldsymbol{G} = \boldsymbol{C}_1 \odot (\boldsymbol{C}_2 \odot (\cdots (\boldsymbol{C}_{k-1} \odot (\boldsymbol{C}_k \odot \boldsymbol{G}))) \cdots).$$

For $i = 1, \ldots, k$ and $j = 1, \ldots, r$, let $\boldsymbol{e}_{i,j}^T$ be the j-th row of $\boldsymbol{E}_i$, and $\boldsymbol{e}_j^T := [\boldsymbol{e}_{1,j}^T \parallel \boldsymbol{e}_{2,j}^T \parallel \cdots \parallel \boldsymbol{e}_{k,j}^T] \in \mathbb{Z}^{1\times kN}$. Then the noise matrix of $\boldsymbol{C}$ has in the j-th row independent subgaussian entries with parameter $O(\|\boldsymbol{e}_j\|_2)$.

Proof. The noise matrix of $\boldsymbol{C}$ is of the form $\sum_{i=1}^k \boldsymbol{E}_i \boldsymbol{X}_i$, where $\boldsymbol{X}_i$ is the matrix used in the evaluation of each $\odot$. By Proposition 3.4, the elements of $\boldsymbol{E}_i \boldsymbol{X}_i$ in the j-th row are independent and subgaussian with parameter $O(\|\boldsymbol{e}_{i,j}\|_2)$. From the Pythagorean additivity of subgaussian random variables, $\sum_{i=1}^k \boldsymbol{E}_i \boldsymbol{X}_i$ has in the j-th row independent subgaussian entries with parameter $O(\|\boldsymbol{e}_j\|_2)$. □

Lemma 2.1, Proposition 3.4, and Corollary 3.1 give a bound for noise in the ciphertext on which SIMD homomorphic operations are evaluated. We can estimate the appropriate modulus q from the bound and Proposition 3.1.

3.2 Switching Plaintext Slots

We constructed a SIMD variant of [GSW13] that supports SIMD homomorphic addition and multiplication. The above scheme, however, is still insufficient to bring out the potential of our scheme.

SIMD homomorphic encryption can encrypt multiple messages: its ciphertext has multiple *plaintext slots*. Gentry et al. [GHS12] showed that any arithmetic circuit can be efficiently computed by the network with gates r-Add, r-Mult, and r-Permute (discussed in Section 1.2). r-Add and r-Mult have already been implemented in our scheme. We here instantiate the plaintext-slot permutation procedure for r-Permute by utilizing the key-switching technique [BV11b, BGV12] similar to [BGH13] and a slightly non-trivial twisting. A rough sketch to construct the plaintext-slot switching procedure is given in Section 1.2. Our switching consists of two algorithms, SwitchKeyGen and SlotSwitch.

- SwitchKeyGen$(\boldsymbol{S}, \pi)$: Given a secret key matrix $\boldsymbol{S} \in \mathbb{Z}_q^{r\times(n+r)}$ for our scheme and permutation π, compute gadgets to switch plaintext slots. Parse $\boldsymbol{S} = [\boldsymbol{I}_r \parallel -\boldsymbol{S}']$, sample $\boldsymbol{A}' \overset{U}{\leftarrow} \mathbb{Z}_q^{n\times N}$ and $\boldsymbol{E}^{(\pi)}, \boldsymbol{E}^{(\pi^{-1})} \overset{R}{\leftarrow} \chi^{r\times N}$, and set

$$\boldsymbol{W}_\pi := \left[\left(\begin{array}{c} \boldsymbol{S}'\boldsymbol{A}' + \boldsymbol{E}^{(\pi)} \\ \hline \boldsymbol{A}' \end{array}\right) + \left(\begin{array}{c} \pi(\boldsymbol{S})\boldsymbol{G} \\ \hline \boldsymbol{0} \end{array}\right) \right]_q \in \mathbb{Z}_q^{(n+r)\times N},$$

$$\boldsymbol{W}_{\pi^{-1}} := \left[\left(\begin{array}{c} \boldsymbol{S}'\boldsymbol{A}' + \boldsymbol{E}^{(\pi^{-1})} \\ \hline \boldsymbol{A}' \end{array}\right) + \left(\begin{array}{c} \pi^{-1}(\boldsymbol{S})\boldsymbol{G} \\ \hline \boldsymbol{0} \end{array}\right) \right]_q \in \mathbb{Z}_q^{(n+r)\times N}.$$

  Output the switch key $\mathsf{ssk}_\pi := (\boldsymbol{W}_\pi, \boldsymbol{W}_{\pi^{-1}})$.

- $\mathsf{SlotSwitch}_{\mathsf{ssk}_\pi}(\boldsymbol{C})$: Take as input a switch key ssk_π for permutation π and ciphertext $\boldsymbol{C}$, output

$$\boldsymbol{C}_\pi \overset{R}{\leftarrow} \boldsymbol{W}_\pi \odot (\boldsymbol{C} \odot (\boldsymbol{W}_{\pi^{-1}} \odot \boldsymbol{G})),$$

where $\boldsymbol{G} \in \mathbb{Z}^{(n+r)\times N}$ is the fixed encryption of $(1, \dots, 1)$ with noise zero.

Since matrices $\pi(\boldsymbol{S})$ and $\pi^{-1}(\boldsymbol{S})$ in key switching gadgets are obtained from interchanging the rows of $\boldsymbol{S}$, we cannot directly apply the DLWE assumption to prove the security. The way of dealing with this problem is to assume the scheme circularly secure.

One nice feature of our plaintext-slot switching is not to suffer from the inconvenience of the security as in [BGH13]: we do not have to use a larger modulus than the underlying encryption scheme. Brakerski et al. [BGH13] made use of a larger modulus $Q = 2^\ell q$, to suppress the noise growth by multiplying a ciphertext to a key switching gadget, so the security of the [BGH13]'s plaintext-slot switching must be related to Q. The larger modulus leads to a higher modulus-to-noise ratio. To obtain the same security level as the underlying [BGH13]'s SIMD scheme, it is required to select a larger dimension. As opposed to this, our plaintext-slot switching can use the same modulus in our scheme.

Correctness and the noise estimation are stated in the following.

Proposition 3.5. *For any permutation π and secret key matrix $\boldsymbol{S}$, let $\mathsf{ssk}_\pi \overset{R}{\leftarrow} \mathsf{SwitchKeyGen}(\boldsymbol{S}, \pi)$. For ciphertext $\boldsymbol{C}$ of our scheme, let $\boldsymbol{C}_\pi \overset{R}{\leftarrow} \mathsf{SlotSwitch}_{ssk_\pi}(\boldsymbol{C})$. Then, we have*

$$\boldsymbol{S}\boldsymbol{C}_\pi = \left(\begin{array}{c} m_{\pi(1)} \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_{\pi(r)} \cdot \boldsymbol{s}_r^T \end{array}\right)\boldsymbol{G} + \hat{\boldsymbol{E}},$$

where $\hat{\mathbf{E}}$ is the noise matrix. In particular, the i-th row of $\hat{\mathbf{E}}$ has independent subgaussian entries with parameter $O(\|[(\boldsymbol{e}_i^{(\pi)})^T \parallel (\boldsymbol{e}_{\pi(i)}^{(\boldsymbol{C})})^T \parallel (\boldsymbol{e}_{\pi(i)}^{(\pi^{-1})})^T]\|_2)$, where $\boldsymbol{e}_i^{(\pi)}$, $\boldsymbol{e}_i^{(\boldsymbol{C})}$, and $\boldsymbol{e}_i^{(\pi^{-1})}$ are the i-th rows of the noise matrices in $\boldsymbol{W}_\pi$, $\boldsymbol{C}$, and $\boldsymbol{W}_{\pi^{-1}}$, respectively.

Proof. We parse the secret key matrix as $\boldsymbol{S} = [\boldsymbol{I}_r \parallel -\boldsymbol{S}']$, where $\boldsymbol{S}' \in \mathbb{Z}^{r\times n}$. Then it holds that

$$\begin{aligned}
\boldsymbol{S}\boldsymbol{C}_\pi &= \boldsymbol{S}\boldsymbol{W}_\pi\boldsymbol{G}^{-1}(\boldsymbol{C}\boldsymbol{G}^{-1}(\boldsymbol{W}_{\pi^{-1}}\boldsymbol{G}^{-1}(\boldsymbol{G}))) \\
&= (\boldsymbol{E}^{(\pi)} + \pi(\boldsymbol{S}')\boldsymbol{G})\boldsymbol{G}^{-1}(\boldsymbol{C}\boldsymbol{G}^{-1}(\boldsymbol{W}_{\pi^{-1}}\boldsymbol{G}^{-1}(\boldsymbol{G}))) \\
&= \widehat{\boldsymbol{E}^{(\pi)}} + \left\{\pi(\boldsymbol{E}^{(\boldsymbol{C})}) + \left(\begin{array}{c} m_{\pi(1)} \cdot \boldsymbol{s}_{\pi(1)}^T \\ \hline \vdots \\ \hline m_{\pi(r)} \cdot \boldsymbol{s}_{\pi(r)}^T \end{array}\right)\boldsymbol{G}\right\}\boldsymbol{G}^{-1}(\boldsymbol{W}_{\pi^{-1}}\boldsymbol{G}^{-1}(\boldsymbol{G})) \\
&= \widehat{\boldsymbol{E}^{(\pi)}} + \pi(\widehat{\boldsymbol{E}^{(\boldsymbol{C})}}) + \left\{\pi(\boldsymbol{E}^{(\pi^{-1})}) + \left(\begin{array}{c} m_{\pi(1)} \cdot \boldsymbol{s}_{\pi^{-1}(\pi(1))}^T \\ \hline \vdots \\ \hline m_{\pi(r)} \cdot \boldsymbol{s}_{\pi^{-1}(\pi(r))}^T \end{array}\right)\boldsymbol{G}\right\}\boldsymbol{G}^{-1}(\boldsymbol{G}) \\
&= \widehat{\boldsymbol{E}^{(\pi)}} + \pi(\widehat{\boldsymbol{E}^{(\boldsymbol{C})}}) + \pi(\widehat{\boldsymbol{E}^{(\pi^{-1})}}) + \left(\begin{array}{c} m_{\pi(1)} \cdot \boldsymbol{s}_1^T \\ \hline \vdots \\ \hline m_{\pi(r)} \cdot \boldsymbol{s}_r^T \end{array}\right)\boldsymbol{G},
\end{aligned}$$

where $\widehat{\boldsymbol{E}^{(\pi)}} = \boldsymbol{E}^{(\pi)}\boldsymbol{G}^{-1}(\boldsymbol{C}\boldsymbol{G}^{-1}(\boldsymbol{W}_{\pi^{-1}})), \pi(\widehat{\boldsymbol{E}^{(\boldsymbol{C})}}) = \pi(\boldsymbol{E}^{(\boldsymbol{C})})\boldsymbol{G}^{-1}(\boldsymbol{W}_{\pi^{-1}})$, and $\pi(\widehat{\boldsymbol{E}^{(\pi^{-1})}}) = \pi(\boldsymbol{E}^{(\pi^{-1})})\boldsymbol{G}^{-1}(\boldsymbol{G})$. If we set $\hat{\boldsymbol{E}} := \widehat{\boldsymbol{E}^{(\pi)}} + \pi(\widehat{\boldsymbol{E}^{(\boldsymbol{C})}}) + \pi(\widehat{\boldsymbol{E}^{(\pi^{-1})}})$, the statement holds from the subgaussian properties and Lemma 2.2. □

The following corollary states how much noise in a ciphertext is amplified by successive applications of SlotSwitch. Similar to the proof of Corollary 3.1, this can immediately be proven from Proposition 3.5.

Corollary 3.2. *For $i = 1, \dots, k$, let π_i be a permutation and $\mathsf{ssk}_i \overset{R}{\leftarrow} \mathsf{SlotSwitch}_{\mathsf{ssk}_i}(\mathsf{sk}, \pi_i)$. Let $\boldsymbol{C} \in \mathbb{Z}^{(n+r)\times N}$ be a packed ciphertext that encrypts a message $\boldsymbol{m} \in \{0,1\}^r$ and has a noise matrix $\boldsymbol{E}^{(\boldsymbol{C})} \in \mathbb{Z}^{r\times N}$, and let*

$$\begin{aligned}
\hat{\boldsymbol{C}} &\overset{R}{\leftarrow} \mathsf{SlotSwitch}_{\mathsf{ssk}_1}(\mathsf{SlotSwitch}_{\mathsf{ssk}_2}(\dots(\mathsf{SlotSwitch}_{\mathsf{ssk}_k}(\boldsymbol{C}))\dots) \\
&= \boldsymbol{W}_{\pi_1} \odot (\dots(\boldsymbol{W}_{\pi_k} \odot (\boldsymbol{C} \odot (\boldsymbol{W}_{\pi_k^{-1}} \odot (\dots(\boldsymbol{W}_{\pi_1^{-1}} \odot \boldsymbol{G}))\dots).
\end{aligned}$$

Then $\hat{\mathbf{C}}$ encrypts $\bigcirc_{i=1}^k \pi_i(\boldsymbol{m})$ and has a noise matrix $\hat{\mathbf{E}} \in \mathbb{Z}^{r\times N}$. In particular, the $j \in [r]$-th row of $\hat{\mathbf{E}}$ has independent subgaussian entries with parameter $O(\|\hat{\boldsymbol{e}}_j\|_2)$, where

$$\begin{aligned}
\hat{\boldsymbol{e}}_j^T := [(\boldsymbol{e}_j^{(\pi_1)})^T \parallel (\boldsymbol{e}_{\pi_1(j)}^{(\pi_2)})^T \parallel \cdots \parallel (\boldsymbol{e}_{\bigcirc_{i=1}^k \pi_i(j)}^{(\boldsymbol{C})})^T \\
\parallel (\boldsymbol{e}_{\bigcirc_{i=1}^k \pi_i(j)}^{(\pi_k^{-1})})^T \parallel \cdots \parallel (\boldsymbol{e}_{\pi_1(j)}^{(\pi_1^{-1})})^T] \in \mathbb{Z}^{1\times(2k+1)N}.
\end{aligned}$$

References

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster Bootstrapping with Polynomial Error. In *CRYPTO*, pages 297–314, 2014.

[Bar10] Boaz Barak. Cryptography course - Lecture Notes, COS 433. Princeton University, 2010. Available at `http://www.cs.princeton.edu/courses/archive/spring10/cos433`.

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed Ciphertexts in LWE-based Homomorphic Encryption. In *PKC*, pages 1–13, 2013.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) Fully Homomorphic Encryption without Bootstrapping. In *ITCS*, pages 309–325, 2012.

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *CRYPTO*, pages 868–886, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *FOCS*, pages 97–106, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Depedent Messages. In *CRYPTO*, pages 505–524, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-Based FHE as Secure as PKE. In *ITCS*, pages 1–12, 2014.

[CCK$^+$13] Jung Hee Cheon, Jean-Sébastian Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch Fully Homomorphic Encryption over the Integers. In *EUROCRYPT*, pages 315–335, 2013.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *EUROCRYPT*, pages 24–43, 2010.

[Gen09a] Craig Gentry. *A FULLY HOMOMORPHIC ENCRYPTION SCHEME*. PhD thesis, Stanford University, Available at `http://crypto.stanford.edu/craig`, 2009.

[Gen09b] Craig Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *STOC*, pages 169–178, 2009.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully Homomorphic Encryption with Polylog Overhead. In *EUROCRYPT*, pages 465–482, 2012.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In *CRYPTO*, pages 75–92, 2013.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *STOC*, pages 1219–1234, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *EUROCRYPT*, pages 700–718, 2012.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A Framework for Efficient and Composable Oblivious Transfer. In *CRYPTO*, pages 554–571, 2008.

[Reg05] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *STOC*, pages 84–93, 2005.

[Rot11] Ron Rothblum. Homomorphic Encryption: from Private-Key to Public-Key. In *TCC*, pages 219–234, 2011.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In *PKC*, pages 420–443, 2010.

[Ver12] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In *Compressed Sensing, Theory and Applications*, pages 210–268. 2012. Available at `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`.