HEMTH: Small Depth Multilevel Thresholding for a Homomorphically Encrypted Image

Paul Nam, Justin Shyi, and Sunwoong Kim
Division of Engineering and Mathematics, University of Washington, Bothell, WA 98011, USA
Email: {poul0315, shyi123, sunwoong}@uw.edu

***Abstract*—One of the image segmentation techniques, multilevel thresholding, is widely used in many computer vision applications because of its low computational complexity and efficient data representation. When it is used in cyber-physical systems and internet-of-things, a special technique is required to protect the sensitive information in an image. This paper proposes a novel homomorphic encryption (HE)-based multilevel thresholding method. To implement a comparison operation in the HE domain, which is not a basic homomorphic operation, a numerical method is adopted. Our proposed method executes comparison operations in parallel to perform more iterations and increase accuracy. When the number of iterations in the numerical comparison operation is (5, 3), the proposed three-level thresholding method shows an average peak signal-to-noise ratio of 28 dB compared to a conventional non-HE-based method and takes 3 minutes on a PC.**

***Index Terms*—homomorphic encryption, multilevel thresholding, cyber-physical systems, internet-of-things**

I. Introduction

Image segmentation is one of the fundamental techniques to analyze an image [1]–[6]. It is widely used in many image processing and computer vision applications, such as medical imaging, content-based image retrieval, and video surveillance. One of the approaches frequently used for image segmentation uses predefined criteria for region partitioning. A thresholding technique, one of the techniques based on this approach, has been an important tool because it enables small storage space, low required transmission bandwidth, fast processing speed, and region-of-interest highlighting [3], [4].

Thresholding techniques are categorized into global thresholding techniques, which use a global threshold(s) for a whole image, and local adaptive thresholding techniques, which use a locally computed threshold for each pixel or block [5], [6]. Typically, the former technique ensures low computational complexity and easy implementation so has got a lot of attention [4]. If a single global threshold is used, an image is segmented into two different regions. Specifically, pixels of which intensity values are greater than a threshold T are classified as object pixels, and the others are classified as background pixels. On the other hand, a multilevel thresholding technique uses multiple thresholds to segment an image into a single background region and several object regions [1]–[3].

Recently, many internet-connected devices are contained in cyber-physical systems (CPS) and internet-of-things (IoT) [7]. In some systems, a cloud server is additionally used to undertake computational burdens of resource- and/or power-hungry user devices. A thresholding technique in CPS and IoT has a distinct issue that does not exist in the conventional one: *security*. In particular, malicious attackers can see user's private images containing sensitive information on networks and cloud servers [8]. Therefore, a privacy-preserving thresholding technique is necessary.

In this paper, a homomorphic encryption (HE) scheme is applied to a multilevel thresholding technique. A comparison operation in thresholding, which is not supported in HE schemes, is implemented using a numerical method. To avoid depth accumulation by successive comparison operations of multilevel thresholding, our proposed HE-based multilevel thresholding (HEMTH) first executes multiple comparison operations in parallel and then merges the results to produce a segmented image in the HE domain. To the best of the authors' knowledge, this is the first paper that applies a HE scheme to an image thresholding technique.

II. Homomorphic Encryption

HE allows operations to be performed on encrypted data without decryption [9]. This helps a public cloud server process data while preserving user's private information. Compared to solutions using other cryptography techniques, an end-to-end HE-based solution reduces the burden on a client because it does not require frequent communications between server and client after sending encrypted data. Thanks to these properties, HE has been applied to image processing algorithms, such as filtering, edge detection, sharpening, and brightness control [8], [10].

Today, Brakerski-Fan-Vercauteren (BFV)/Brakerski-Gentry-Vaikuntanathan (BGV) [11]–[13], Cheon-Kim-Kim-Song (CKKS) [14], and Ducas-Micciancio (DM)/Chillotti-Gama-Georgieva-Izabachène (CGGI) [15], [16] HE schemes are widely used. They are implemented in open-source libraries, such as Microsoft SEAL [17]. Their plaintext data types are different. For example, the BFV/BGV, CKKS, and DM/CGGI schemes deal with integer, real number, and boolean plaintext messages, respectively.

A. Basic Homomorphic Operations

The HE schemes for integer and real number plaintext messages provide the following homomorphic operations [9]:

- `HomAdd`: ciphertext-ciphertext addition;
- `HomAddPlain`: ciphertext-plaintext addition;
- `HomSub`: ciphertext-ciphertext subtraction;
- `HomSubPlain`: ciphertext-plaintext subtraction;

978-1-6654-7189-3/22/$31.00 © 2022 IEEE

Algorithm 1 Comp$(x, y; n, d)$ [18]

Input: normalized real numbers $x, y \in [0, 1]$
Input: the number of iterations $n, d \in \mathbb{N}$
Output: a value between 0 and 1

1: $a \leftarrow x - y$
2: **for** $(i = 1; i \leq d; i = i + 1)$ **do**
3: $\quad a \leftarrow f_n(a)$
4: **end for**
5: **return** $(a + 1)/2$

- HomMult: ciphertext-ciphertext multiplication;
- HomMultPlain: ciphertext-plaintext multiplication.

The results of these operations are ciphertexts. Unlike the other homomorphic operations, HomMult includes a key switching operation, also called the relinearization, to restore the deformed key. Furthermore, the CKKS scheme requires an additional operation called rescaling to control the precision and minimize the growth of error.

Each time a homomorphic operation is performed, the magnitude of noise used to hide plaintext messages increases. In particular, HomMult nearly doubles the magnitude. If the noise level exceeds a certain level, decryption does not guarantee desirable results. Therefore, a limited number of consecutive HomMults are allowed for homomorphic evaluation. This number is called the multiplicative circuit depth, or just the *depth*. Note that there is a technique called bootstrapping that makes the depth infinite, but it requires tremendous computational complexity. Therefore, the bootstrapping technique is not considered in this paper.

Many recent HE schemes support packing techniques. With these techniques, multiple plaintext messages are packed into a single vector, and this vector is transformed into a ciphertext. A homomorphic operation on this ciphertext corresponds to element-wise or single instruction/multiple data (SIMD) operations in the plaintext domain. Therefore, the execution time is significantly improved.

B. Comparison of Homomorphically Encrypted Data

HE schemes for integer and real number plaintext messages do not support a homomorphic comparison operation by default. To solve this problem, Cheon *et al.* proposed a numerical method for a comparison operation in the CKKS scheme [18]. Algorithm 1 describes this method. This algorithm starts with the following formula, comparison result $= \frac{f(x-y)+1}{2}$, where $f(\cdot)$ is the step function that produces 1 if $x > y$; -1 if $x < y$; and 0 otherwise. Since this step function is non-linear, it is approximated as follows:

$$f_n(a) = \sum_{j=0}^{n} \frac{1}{4^j} \cdot \binom{2j}{j} \cdot a(1 - a^2)^j. \quad (1)$$

Algorithm 1 implemented in the HE domain requires a large depth because it includes a nested loop, in which an inner loop has n iterations and an outer loop has d iterations. Ideally, the depth of Algorithm 1, denoted as $\text{depth}_{\text{comp}}$ in the rest of this paper, is approximately $d\lceil \log_2 n \rceil$. Fig. 1 shows how the depth is accumulated in $f_n(a)$ when n is 4. The constant in this figure stands for $\frac{1}{4^j} \cdot \binom{2j}{j}$, which is calculated in advance, and intermediate results, such as $(1 - a^2)^2$, are reused to avoid unnecessary computations. When operands with different depths are multiplied by each other, a smaller depth is pulled up to a larger depth before the operation. Therefore, the total depth in $f_n(a)$ grows logarithmically.

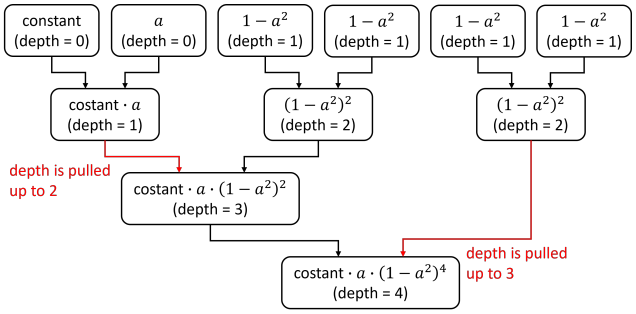
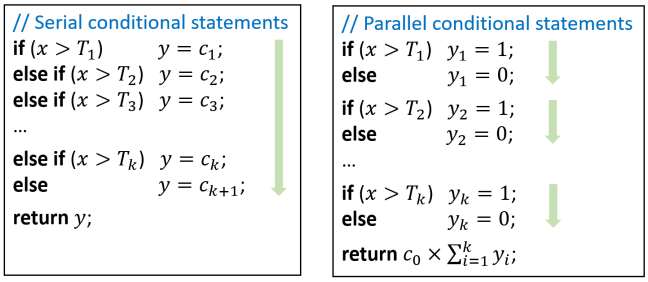


Fig. 1. Depth accumulation in $f_4(a)$ calculation.



(a) (b)

Fig. 2. Pseudo code for multilevel thresholding in the non-HE domain. x is an input pixel intensity value and T_i is the i-th threshold. c_i in Fig. 2(a) is the i-th potential pixel intensity value, and c_0 in Fig. 2(b) is the base potential pixel intensity value. (a) naïve method (b) proposed method for HEMTH.

III. Privacy-Preserving Multilevel Thresholding

This section presents our proposed HEMTH that adopts Algorithm 1 for multilevel thresholding in the HE domain. Note that the numerical comparison operation is denoted as the Comp operation in the rest of this paper to distinguish it from the conventional comparison operation. In addition, for the sake of simplicity, the same n and d values are used for different Comp operations working together.

A. Small Depth Multilevel Thresholding

Typically, a thresholding technique includes a comparison operation(s). In particular, for multilevel thresholding, multiple comparison operations should be performed. The most naïve method for multilevel thresholding is to use if ... else if ... else statements, as shown in Fig. 2(a). However, when moved to the HE domain, this method performs Comp operations in serial, which incurs a large depth. For example, the total depth in three-level thresholding is $2 \times \text{depth}_{\text{comp}} + 1$, which requires long latency or even may not satisfy a depth constraint. This

problem gets worse as the number of levels in thresholding increases.

To solve this problem, HEMTH performs Comp operations in parallel and merges the results, which greatly lowers the total depth. Fig. 2(b) shows its pseudo code in the non-HE domain. It replaces the if ... else if ... else statements of the naïve method with independent if ... else statements. Suppose that there are three levels in thresholding, and r_1 and r_2 stand for $\texttt{Comp}(x, T_1, n, d)$ and $\texttt{Comp}(x, T_2, n, d)$, respectively. T_1 and T_2 are thresholds ($T_1 > T_2$). Additionally, suppose that three potential pixel intensity values are evenly distributed between the maximum and minimum values expressed with a given bits per pixel (bpp), which is different from the naïve method that selects any potential pixel intensity values. A segmented pixel intensity y is then calculated as follows:

$$y = (r_1 + r_2) \cdot \lfloor \frac{2^{bpp} - 1}{2} \rfloor. \tag{2}$$

For (k+1)-level thresholding, this equation is generalized as follows:

$$y = \Sigma_{i=1}^{k} r_i \cdot \lfloor \frac{2^{bpp} - 1}{k} \rfloor, \tag{3}$$

where $r_i = \texttt{Comp}(x, T_i, n, d)$ and T_i is the i-th threshold.

Since Comp operations in HEMTH are executed in parallel, the total depth is $\text{depth}_{\text{comp}}$ regardless of the number of levels. For three-level thresholding, HEMTH reduces the depth by approximately half compared to the naïve method, and the improvement increases as the number of levels increases.

Unlike the conventional thresholding method that exactly outputs a fixed number of potential pixel intensity values, HEMTH outputs approximate potential pixel intensity values. Due to the characteristics of the numerical Comp operation, an image segmented by HEMTH becomes closer to an image segmented by the conventional method as the n and d values increase. In fact, segmenting an image in the HE domain does not help reduce data size nor improve execution time, but it can be used to highlight regions of interest in an encrypted image. Accuracy in this application is not significantly affected by the approximate technique, which is described in Section IV-C.

B. Overall Flow

In this paper, HEMTH is implemented as a client-server model. Fig. 3 shows a high-level overall flow of our proposed design. First of all, an input image is read on the client-side. Suppose that all pixels in this image cannot be encoded and encrypted into a single ciphertext because the image size is too large (the number of elements in a vector or slots in a ciphertext is insufficient to contain all pixels). To create multiple ciphertexts, the input image is split into several blocks, and each block is encoded and encrypted into a separate ciphertext. The block size is determined based on the number of slots in a ciphertext, and this number depends on HE parameters. The details are explained in Section IV-B. The generated ciphertexts are sent to the server through networks, and HEMTH segments the encrypted image. Since the resulting image is still encrypted, the server cannot acquire any sensitive information in the image. After being sent from the server to the client, the resulting image blocks are decrypted, decoded, and merged, and finally, the client gets the segmented image.

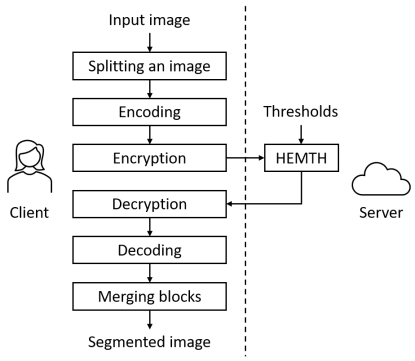


Fig. 3. Overall flow of our proposed design.

TABLE I
HE PARAMETERS USED FOR HEMTH

Security level	N	log$q$	Maximum depth†
128-bit	2^{15}	885	20

†When 40 bits are allocated to all primes.

IV. EVALUATION

A. Experimental Setup

In HEMTH implementation, Microsoft SEAL version 3.6 is used [17]. Among the HE schemes available in this library, the CKKS scheme is used for the Comp operation using real numbers. Generally, HE parameters greatly affect performance in homomorphic evaluation. There are two critical parameters: polynomial degree N and total bit-length of q (log$q$). N is a power of two, and q is a product of different primes. In the CKKS scheme, one of the primes is dropped whenever HomMult is performed. Therefore, the number of primes determines the maximum available depth in homomorphic evaluation. For the same security level, N and log$q$ are proportional to each other [9]. Therefore, to increase the maximum available depth, the N value needs to be increased. However, as the N value increases, the execution time increases significantly. It makes homomorphic evaluation impractical, so the maximum N value in SEAL is limited to 2^{15}.

This paper aims for a 128-bit security level, which is common in recent real-world applications, and uses the N value of 2^{15} to make a sufficiently large maximum available depth. According to the SEAL manual, the bit-length of q corresponding to this N value is 885 bits [19]. If 40 bits are allocated to each prime, 22 (=$\lfloor 885/40 \rfloor$) primes are generated. Since both end primes are used for special purposes, the maximum available depth is 20. Table I summarizes the HE parameters used in this paper.

As input images, six 512×512-sized grayscale standard images with a bpp of 8, shown in Fig. 4, are used [20]. In this evaluation section, the number of thresholding levels is

(a) (b) (c)

(d) (e) (f)

Fig. 4. Test input images (512×512 grayscale). (a) Boat (b) Cameraman (c) Lake (d) Peppers (e) Pirate (f) Woman.

set to 3, and each segmented pixel intensity value is (approximately) 0, 127, or 254. For comparison, a conventional three-level thresholding implementation using if ... else if ... else statements is used as a baseline. In addition, another non-HE-based implementation that replaces the conventional comparison operation with the `Comp` operation is used. This implementation is referred to as "non-HE" in Table II. Note that there is no previous work on HE-based thresholding techniques, so only results by HEMTH are presented for the HE domain. As hardware for our proof-of-concept design, a PC with the Intel Xeon W-2295 and 128GB RAM is used. The operating system is Ubuntu 18.04 LTS.

B. Encoding

Given the N value of 2^{15}, each ciphertext of the CKKS scheme has $2^{15}/2$ slots [21]. In other words, 2^{14} real numbers can be packed into a single ciphertext maximally. If all slots in a ciphertext are used, a 512×512-sized grayscale image generates 16 ciphertexts. In our implementation, a 512×512-sized image is divided into 16 128×128-sized blocks, and each block is encoded and encrypted into each ciphertext.

C. Accuracy

To evaluate the accuracy of images segmented by HEMTH, peak signal-to-noise ratio (PSNR) is used. PSNR and mean squared error (MSE) values are calculated using (4) and (5), respectively.

$$\text{PSNR (dB)} = 10 \cdot \log_{10}((2^{bpp}-1)^2/\text{MSE}), \tag{4}$$

$$\text{MSE} = \frac{\Sigma_{i=0}^{R-1}\Sigma_{j=0}^{C-1}[g(i,j)-h(i,j)]^2}{R \times C}, \tag{5}$$

where R and C are the height and width of an image, and $g(i,j)$ and $h(i,j)$ are pixel intensity values segmented by the conventional thresholding technique using if ... else if ... else statements and the modified thresholding techniques using `Comp` operations at a coordinate (i,j).

Table II compares the PSNR values of the three-level thresholding techniques. The first and second columns show the d and n values of the `Comp` operation, respectively. The third column shows the total depth required for thresholding. Note that this table shows only the results when the total depth is smaller than the maximum available depth shown in Table I. In addition, d = 2 cases showing relatively low PSNR values and $n \geq 8$ cases requiring long execution time are not included due to limited space. Theoretically, each `HomMultPlain` does not necessarily require depth consumption. However, in SEAL, a depth is consumed even after `HomMultPlain` to provide ease of use without worrying about noise accumulation [22]. Therefore, the depth results in Table II do not exactly follow $\text{depth}_{\text{comp}}$. However, the depth is still maintained until the value of n becomes a power of 2.

Columns 4-15 in Table II show the PSNR results for the six text images. First of all, the non-HE and HEMTH designs incur PSNR differences because the CKKS scheme performs rounding off to plaintext messages in a ciphertext after each homomorphic multiplication. However, the PSNR difference is 0.1% on average, which is negligible. Depending on the test image, d value, and n value, HEMTH occasionally shows higher PSNR results than the non-HE design. It is due to probability and does not mean that HEMTH outperforms the non-HE design in terms of accuracy.

In all the six images, if the d value is fixed, the PSNR value increases as the n value increases. Similarly, if the n value is fixed, the PSNR value increases as the d value increases. These results are in line with what is argued by Cheon *et al.* [18]: $f_n(a)$ in the outer loop with d iterations becomes closer to the ideal step function as the n and d values increase. In fact, the PSNR value increases in a logarithmic fashion as the n value increases, while the execution time increases approximately linearly. Therefore, it is necessary to consider the increase in PSNR and computational complexity together to find the optimal parameter values, which is described in Section IV-D.

Fig. 5 shows the images segmented by the conventional and proposed three-level thresholding designs. Here, the n value is fixed, and the d value changes from 3 to 5. Specifically, the n value is set to 3, which is the maximum value available for the d value of 5 under the depth constraint. As described in Section III-A, the `Comp` operation outputs not exact 0 and 1 but approximate values, so the images segmented by HEMTH contain more than three values. However, as the d value (and the n value) increases, they get close to the images segmented by the conventional design.

D. Execution Time

In this subsection, execution time is evaluated. Fig. 6(a) shows the execution time of HEMTH. The horizontal and vertical axes refer to the n value and execution time in a minute, respectively, and the three line graphs correspond to different d values. For all the d values, the execution time increases as the n value increases. Overall, the three line graphs show linear trends. However, when the n value increases from 3 to 4, where the depth changes, the slopes of the graphs for

TABLE II
PSNR COMPARISON OF THE THREE-LEVEL THRESHOLDING TECHNIQUES

d	n	depth	PSNR (dB)											
			Boat		Cameraman		Lake		Peppers		Pirate		Woman	
			non-HE	HE†	non-HE	HE†	non-HE	HE†	non-HE	HE†	non-HE	HE†	non-HE	HE†
3	2	12	20.24	20.27	19.60	19.59	18.92	18.89	18.92	18.90	18.84	18.84	19.83	19.83
3	3	12	22.05	22.07	21.36	21.33	20.82	20.78	20.83	20.77	20.50	20.50	21.61	21.60
3	4	15	23.77	23.79	23.09	23.09	22.64	22.62	22.51	22.52	22.05	22.06	23.19	23.19
3	5	15	25.06	25.06	24.44	24.42	24.09	24.04	23.91	23.85	23.30	23.29	24.45	24.43
3	6	15	26.06	26.14	25.46	25.53	25.19	25.22	25.02	25.07	24.32	24.39	25.45	25.52
3	7	15	26.99	27.03	26.40	26.43	26.17	26.17	26.06	26.09	25.27	25.30	26.39	26.41
4	2	15	22.83	22.91	22.14	22.18	21.63	21.65	21.59	21.60	21.19	21.25	22.32	22.37
4	3	15	25.70	25.70	25.09	25.08	24.79	24.75	24.59	24.59	23.94	23.94	25.08	25.08
4	4	19	27.72	27.72	27.14	27.13	26.94	26.90	26.90	26.87	26.03	26.03	27.13	27.12
4	5	19	29.32	29.32	28.74	28.73	28.60	28.56	28.70	28.67	27.70	27.69	28.75	28.74
4	6	19	30.65	30.62	30.07	30.04	29.96	29.91	30.25	30.19	29.09	29.07	30.08	30.04
4	7	19	31.70	31.70	31.12	31.12	31.04	31.02	31.66	31.69	30.21	30.21	31.13	31.12
5	2	18	25.71	25.75	25.10	25.13	24.80	24.80	24.60	24.63	23.95	23.99	25.09	25.12
5	3	18	29.02	29.01	28.44	28.42	28.28	28.24	28.34	28.31	27.38	27.37	28.44	28.42

†Our proposed HEMTH including encoding, encryption, decryption, and decoding processes.

$d = 3$ and $d = 4$ deviate from the linear trend. It is because the bit-length of q that is actually used changes, and thus the execution time for each homomorphic operation increases. The slope increases as the d value increases because the increased execution time in the $f_n(a)$ calculation is accumulated more.

Fig. 6(b) shows the results for all the (d, n) cases depending on the average PSNR and execution time. As the number of iterations increases, the average PSNR value and execution time increase together. However, while the execution time increases almost linearly, the increase in average PSNR gradually decreases, which lowers the efficiency. However, since a low PSNR due to a small number of iterations may not be acceptable, it may be useful to find the optimal (d, n) values in partitioned PSNR sections. For example, high efficiency can be obtained by using (d, n) of (4, 2) in the 20-25 dB section. When a PSNR value greater than 25 dB is required, using (d, n) of (5, 3) leads to high efficiency.

The non-HE-based implementation using the `Comp` operation shows 179× faster average processing speed on the same machine. The execution time of HEMTH was measured using a single thread. Therefore, it can be improved by using multiple threads. In addition, using custom hardware designs of homomorphic operators can help with acceleration [23], [24].

V. CONCLUSION

This paper proposes a privacy-preserving multilevel thresholding method. It focuses on how to do multilevel thresholding in the HE domain, rather than on how to find the best threshold. Specifically, to solve the depth problem, multiple comparison operations are executed in parallel. This makes multilevel thresholding use the same depth as bi-level thresholding in the HE domain, which in turn increases the number of available iterations and the accuracy. To the best of the authors' knowledge, this is the first paper on multilevel thresholding in the HE domain. For future work, we are working on accelerating HEMTH using FPGA-based custom hardware designs.

ACKNOWLEDGMENT

We thank Dr. Rob Rutenbar, Dr. Jung Hee Cheon, Wonhee Cho, and Keewoo Lee for their continued support.

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00840, Development and Library Implementation of Fully Homomorphic Machine Learning Algorithms supporting Neural Network Learning over Encrypted Data). In addition, this material is based upon work supported by the National Science Foundation under Grant No. 2105373.

REFERENCES

[1] P.-S. Liao, T.-S. Chen, and P.-C. Chung, "A Fast Algorithm for Multilevel Thresholding," *J. Inf. Sci. Eng.*, vol. 17, no. 5, pp. 713–727, 2001.

[2] J.-C. Yen, F.-J. Chang, and S. Chang, "A New Criterion for Automatic Multilevel Thresholding," *IEEE Trans. Image Process.*, vol. 4, no. 3, pp. 370-378, Mar. 1995.

[3] S. Arora, J. Acharya, A. Verma, and P. K. Panigrahi, "Multilevel Thresholding for Image Segmentation Through a Fast Statistical Recursive Algorithm," *Pattern Recognit. Lett.*, vol. 29, no. 2, pp. 119–125, 2008.

[4] S. S. Al-Amri, N. V. Kalyankar, and S. D. Khamitkar, "Image Segmentation by Using Threshold Techniques," *J. Computing*, vol. 2, no. 5, pp. 83–86, May 2010.

[5] F. Shafait, D. Keysers, and T. M. Breuel, "Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images," in *Proc. Doc. Recognit. Retriev.*, 2008.

[6] N. Senthilkumaran and S. Vaithegi, "Image Segmentation by Using Thresholding Techniques for Medical Images," *Comput. Sci. Eng., Int. J.*, vol. 6, no. 1, pp. 1-13, Feb. 2016.

[7] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Berkeley, CA, USA: Lee & Seshia, 2011.

[8] M. T. I. Ziad, A. Alanwawr, M. Alzantot, and M. Srivastava, "Cryptoimg: Privacy Preserving Processing Over Encrypted Images," in *Proc. CNS*, 2016.

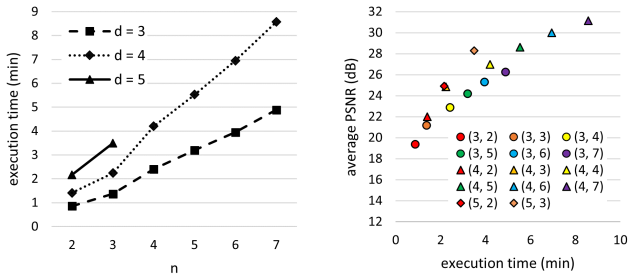
[9] J. H. Cheon *et al.*, "Introduction to Homomorphic Encryption and Schemes," in *Protecting Privacy through Homomorphic Encryption*, Springer, 2021, pp. 3–28.

[10] S. D. Kannivelu and S. Kim, "A Homomorphic Encryption-based Adaptive Image Filter Using Division Over Encrypted Data," in *Proc. RTCSA*, 2021.

[11] Z. Brakerski, "Fully Homomorphic Encryption Without Modulus Switching From Classical GapSVP," in *Proc. CRYPTO*, 2012.

[12] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," Cryptology ePrint Archive, Tech. Rep. 2012/144, 2012.

Fig. 5. Images segmented by the three-level thresholding methods. (a)-(f) conventional (non-HE-based) (g)-(l) HEMTH (d = 3; n = 3) (m)-(r) HEMTH (d = 4; n = 3) (s)-(x) HEMTH (d = 5; n = 3).



(a) (b)

Fig. 6. Execution time evaluation of HEMTH (three-level). (a) execution time vs. n (b) average PSNR vs. execution time.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption Without Bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1-36, 2014.

[14] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," in *Proc. ASIACRYPT*, 2017.

[15] L. Ducas and D. Micciancio, "FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second," in *Proc. EUROCRYPT*, 2015.

[16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast Fully Homomorphic Encryption Over the Torus," *J. Cryptology*, vol. 33, pp. 34-91, 2020.

[17] Microsoft SEAL (release 3.6), [Online]. Available: https://github.com/Microsoft/SEAL, 2020.

[18] J. H. Cheon, D. Kim, and D. Kim, "Efficient Homomorphic Comparison Methods with Optimal Complexity," in *Proc. ASIACRYPT*, 2020.

[19] H. Chen, K. Han, Z. Huang, A. Jalali, and K. Laine, "Simple Encrypted Arithmetic Library v2.3.0," [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/12/sealmanual.pdf

[20] Standard Test Images, [Online]. Available: http://imageprocessingplace.com/root_files_V3/image_databases.htm

[21] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for Approximate Homomorphic Encryption," in *Proc. EUROCRYPT*, 2018.

[22] M. Cho, K. Lee, and S. Kim, "HELPSE: Homomorphic Encryption-based Lightweight Password Strength Estimation in a Virtual Keyboard System," in *Proc. GLSVLSI*, 2022.

[23] S. Kim, K. Lee, W. Cho, J. H. Cheon, and R. A. Rutenbar, "FPGA-based Accelerators of Fully Pipelined Modular Multipliers for Homomorphic Encryption," in *Proc. ReConFig*, 2019.

[24] S. Kim, K. Lee, W. Cho, Y. Nam, J. H. Cheon, and R. A. Rutenbar, "Hardware Architecture of a Number Theoretic Transform for a Bootstrappable RNS-based Homomorphic Encryption Scheme," in *Proc. FCCM*, 2020.