Privacy Preserving Morphological Operations for Digital Images

Cristian Lupaşcu[1,2] , Cezar Pleşca[1,2] and Mihai Togan[1,2]
[1] Research and Innovation, certSIGN, Bucharest, Romania
[2] Military Technical Academy "Ferdinand I", Bucharest, Romania
Contact author e-mail: cristian.lupascu@certsign.ro

***Abstract*—Some image processing operations like segmentation or edge detection usually require complex computations. The emerging cloud technologies provide businesses and individuals with the possibility to outsource these computations on a remote platform with large resources and high availability. This approach substantially reduces the costs, but could rise privacy issues. To address these issues, the user data could be encrypted, sent to the cloud and processed there by taking advantage from the properties of a homomorphic encryption scheme. The result of the computation performed on encrypted data is sent back to the user to be decrypted.**

This paper studies the possibility of conducting some morphological operations on encrypted digital images. First, we model these processing using the two elementary operations in the ring $\mathbb{F}_2$. Second, we implement these computations using CSGNHE which is a recently proposed scheme by our team. Finally, we compare our results in terms of execution times and encrypted data size with some well-known schemes from the literature.

***Index Terms*—homomorphic encryption, morphological operations, privacy preserving, image processing**

I. Introduction

Mathematical morphology belongs to a branch of non-linear image processing and analysis that deals with the geometric structures within a digital image. This approach is based on the assumption that an image consists of geometric structures which may be handled by set theory [1].

Morphological operations are used in many fields of image processing like segmentation (determine the form of the objects inside the image) or edge detection (determine the contour of the objects) [2]. A direct application of image segmentation is automatic text extraction which can be further useful for number plate recognition systems. Morphological operations are also used in some biometrics applications such as fingerprint identification to enhance minutiae quality.

To process a given image, morphological operations use, as a parameter, a structuring element which describes a shape. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels. It is well known that one can obtain the expression of morphological filters in terms of two primitive morphological operations, namely erosion and dilation.

Erosion shrinks or thins the objects in a binary image. The manner or the extent of shrinking is controlled by structuring elements. Dilation is used to increase the area of an object by adding pixels around the boundaries and also filling interior holes inside objects.

Medical image processing can also benefit from segmentation in order to compute some numerical features of a certain shape that could be used to analyze the evolution of that specific form. The set of these features could also serve as a distinct signature in the form recognition process. These features include geometric characteristics such as perimeter, area, compactness or circularity and also invariant statistics moments of different orders.

In the medical imaging field, the need for privacy is a crucial issue when dealing with data remotely processed. An emerging solution to this problem is the use of homomorphic encryption approach, in which the data is first encrypted, then sent to the cloud and the remote computation (e.g. a morphological primitive) is performed on an encrypted image.

This is an instance of the large class of applications that can be realized if the theoretical construction of Fully Homomorphic Encryption Schemes (FHE) would achieve a high level of efficiency (see for example [3]). Fully Homomorphic Encryption is considered to be the holy grail of cryptography, an elusive goal which could solve the IT world's problems of security and trust.

Research in the area started with the groundbreaking work [4] of Gentry in 2009, where it was shown that FHE can be realized in principle. Since that time, considerable progress has been made in finding more practical and more efficient solutions involving novel mathematical techniques, and a number of exciting applications.

Gentry's thesis [4] presents not only the first description of a FHE scheme, but also a powerful framework for achieving FHE. However, it is practically not a realistic scheme because the bootstrapping part, which is the intermediate refreshing procedure of a processed cipher-text, is too costly in terms of computation. For this reason a lot of follow-up work in the last ten years focused on finding improvements of the bootstrapping procedure in new proposed schemes. Such schemes have been implemented:

- HElib [5] - a software library, developed by Halevi and Shoup, and based on the BGV [6] and CKKS [7] schemes.
- libScarab [8] - an implementation of the Fully Homomorphic Smart-Vercauteren scheme [9].
- FHEW [10] - an implementation of Ducas and Micciancio FHE scheme [11].

- TFHE [12] - an implementation of Chillotti et al FHE scheme available [13].
- SEAL [14] - an implementation of Fan and Vercauteren FHE [15] and CKKS [7] schemes.

On the other hand, CSGNHE sheme (shortly CSGN - see section III) is a noise-free FHE that employs a blue-print for creating FHE from an AND- homomorphic encryption scheme. Theoretical performance of this scheme is complementary to that above mentioned FHE schemes in the sense that the CSGNHE scheme performs better on AND-gates while the LWE schemes perform better on XOR-gates.

For this reason, a circuit may be suited for one or the other option of FHE schemes. For example, in the case of morphological primitives applied on images, the main involved operation is the multiplication and the usage of large structural elements will limit the efficiency of the LWE schemes. In the case of such applications a noise free scheme like CSGNHE, which performs well in multiplication operations, shall be more suitable.

This paper presents some comparative results in terms of time and memory consumption for two LWE-based FHE implementations (BGV [6] and BFV [15]) and a noise free FHE scheme (CSGNHE [16]) in the case of image processing applications that requires heavy multiplications.

The rest of the paper is organized as follows: Section 2 briefly presents some aspects regarding some morphological operations applied on the digital images. Section 3 presents a short description for our noise-free scheme CSGNHE. Section 4 presents the conducted experiments and the achieved comparative results. Section 4 draws our conclusion.

II. Morphological Primitives

The term morphology refers to the description of the properties of shape and structure of any objects. In the context of image processing, this term refers to the description of the properties of shapes of areas in one image. Morphological operations were originally defined as operations on sets, but it soon became clear that they are also useful in the processing tasks of the set of points in the two-dimensional space.

In mathematical morphology, the sets are represented by objects in the image. If one adds the set of all background pixels, then one obtains the full description of a binary image. The morphology operators are used to extract some properties of the image, useful for its descriptions (e.g. contours, skeletons or convex hulls). Morphological methods are also used in the chain of operations for the image segmentation.

The input data for the mathematical morphology are two parameters : a binary image and a structural element which describes some geometrical shape. The are many morphological operations but all of them can be expressed using two primitives: erosion and dilation.

A. Binary Images : Objects and Background

In the binary images, the pixels are partitioned in two sets, namely objects and background. An object of a certain shape can be represented by a set of either 4- or 8-connected white pixels in black background. The morphological operations between the structural element and the image are expressed in the form of classical operations on sets (intersection, union, etc.) and therefore, the result is also a set.

Usually, we represent a black and white image as a binary matrix, where value 0 represent the background pixels and value 1 the pixels belonging to objects. Of course, one could choose between this representation and its complementary one (0 assigned to objects and 1 to background). Otherwise, one can formally see the image as the set of pixels that represent the objects; this approach makes the understanding of morphological primitives more easy.

B. Structural Element

A structural element can be regarded as a description of one area with some form. Typically, such an element is much smaller than the processed image. The most common elements have a special name: Box (i.e. a rectangle of given size), Disk (a disk with a given radius), Ring (i.e. a ring of a given size).

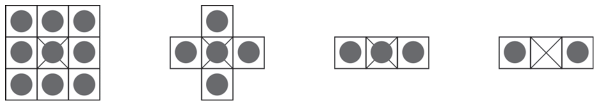
Fig. 1. Four examples of structured elements

Some examples of elements are illustrated in figure 1. All elements s of the structuring element S are measured with respect to the origin $(0, 0)$, marked with an X in the figure. Typically, an element is symmetric, either central or with respect to its vertical or horizontal axis. Generally, the origin of the structured element is chosen to be the pixel located at its center. In the figure 1, the set S for the second structuring element is : $S = \{(0, -1), (-1, 0), (0, 0), (1, 0), (0, 1)\}$.

The translation of a structuring element is an operation used to compute the result of a certain morphological primitive. The translation of a structuring element S in a given position (i.e pixel in the image) $p = (x, y)$ is defined as follows:

$$S_p = \{c \mid c = s + p, \text{ for all } \quad s \in S\}$$

where $c = (c_x, c_y) = (s_x + x, s_y + y) = s + p$.

C. Erosion

The result of the erosion operation applied to an image I (seen as a set of pixels) and a structuring element S is the set of pixels p for which, the translated structured element S_p is completely contained in the image I.

$$E(I, S) = I \ominus S = \{p = (x, y) \mid S_p \subseteq I\}$$

Therefore, the result of the erosion operator retains only the positions in the image for which, all the pixels of the translated structured element belong to the object set. Of course, this computations is done over all positions p where the S_p is entirely contained in the image matrix.

The eroded image is obtained by translating (or sliding) S by p over the image, from top to bottom and from left to right. In general, erosion of a binary image shape I by S shrinks the shape by half of the size of S. One can see the result of the erosion operator on an object shape in figure 2 : the back pixels represent the object and the gray pixels represent the positions removed from the initial object mask [17].

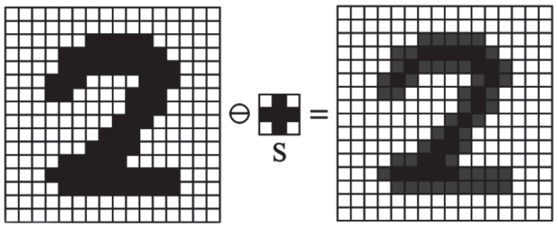

Fig. 2. The effects of erosion with structured element S

Let's note with A the matrix representation of I using the following representation : 1 for objects, 0 for background. To compute the new eroded image (represented by A') in the position p, one must check the value of the statement:

$$\forall s = (x, y) \in S_p \implies A[x][y] = 1$$

If the statement holds, then $p = (i, j)$ is kept in the new image (i.e A'[i][j]=1), otherwise p is removed from the initial image (i.e A'[i][j]=0). It is easy to see that the new value for $p = (i, j)$ can be expressed as a multiplication as follows:

$$A'[i][j] = \prod_{s \in S_p} A[s_x][s_y]$$

One can note that this computation can also be done in an encrypted space if we have a homomorphic encryption scheme with respect to multiplication. We only have to know the encrypted binary image and the structural element in clear.

D. Dilation

Formally, one can define the dilation operator for an image I as the set of pixels p for which, the translated structured element S with p is partially contained in the image.

$$D(I, S) = I \oplus S = \{p = (x, y) \mid S_p \cap I \neq \Phi\}$$

where Φ represents empty set. Therefore, the result of the dilation operator retains only the positions in the image for which, at least one of the pixels of the translated element belong to the object set.

In general, dilation of a binary image shape I by S expands the shape I by half of the size of S. One can see the result of the dilation operator on an object shape in figure 3 : the back pixels represent the object and the gray pixels represent the positions added to the initial object mask.

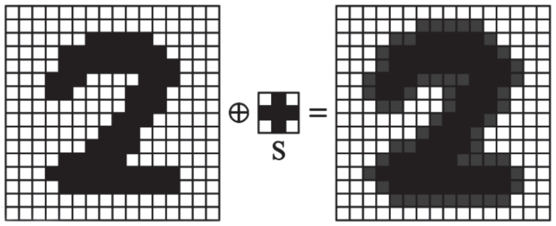

Fig. 3. The effects of dilation with structured element S

We'll keep the same notations as before but represent the image using complementary code, namely : 0 for objects, 1 for background. To compute the new dilated image, one must check the value of the following statement:

$$\exists s = (x, y) \in S_p \mid A[x][y] = 0$$

If the statement holds, then $p = (i, j)$ is kept in the new image (i.e A'[i][j]=0), otherwise p is removed from the initial image (i.e A'[i][j]=1). Again, one can see that the new value for $p = (i, j)$ can be expressed as a multiplication of all the pixels values from S_p.

$$A'[i][j] = \prod_{s \in S_p} A[s_x][s_y]$$

Note that if one keeps the initial representation used to explain erosion primitive (i.e. 0 for objects, 1 for background), then the dilation computation could be done similarly, as follows : complement the pixels values, make the product and then complement the result.

$$A'[i][j] = 1 - \prod_{s \in S_p} \big(1 - A[s_x][s_y]\big)$$

E. Opening and Closing

Using one or both primitives described previously, one can compute other morphology based operations like edge detection. This operations can be realized by simply taking the difference (in the ring $\mathbb{F}_2$) between the initial image and its eroded version, as one can see in figure 4.

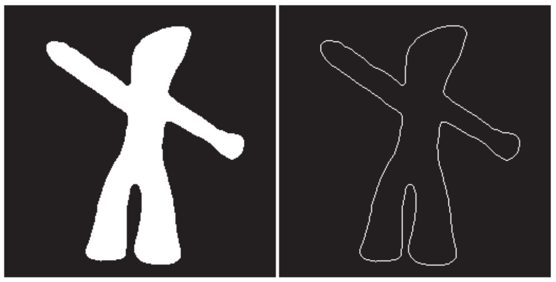

Fig. 4. Edge detection using erosion operator

Another well known morphological operators, namely the opening and the closing, can be expressed in terms of the two primitives. The opening of I and S is the dilation with S' of the erosion of I by S, where S' is the symmetric of S with respect to origin.

$$I \circ S = (I \ominus S) \oplus S' \qquad S' = \{-s | s \in S\}$$

Note that dilation and erosion are not a pair of opposite operations in the sense that their effects do not cancel each other. The erosion carried out first eliminates small shapes (i.e. the noise) as well as shrinking the object shape, while the following dilation grows the object back (but not the noise).

A similar operation, the closing of I and S is the erosion with S' of the dilation of I by S. The dilation carried out first eliminates small holes inside the object shape (assumed to be noise) as well as expanding the object shape, while the following erosion shrinks the object back (but not the noise).

$$I \bullet S = (I \oplus S) \ominus S'$$

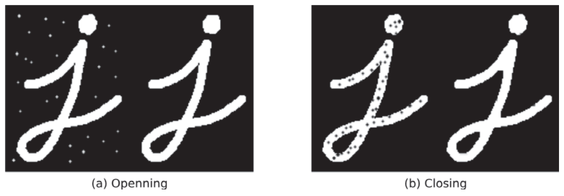

Fig. 5. Opening and Closing Examples

III. certSIGN Encryption Scheme

We start this section by presenting a symmetric monoid homomorphic encryption scheme over $(\mathbb{F}_2, \cdot)$, which was introduced in [16] (see also [18]). We shall present a simplified version of the scheme. For the complete and secure version of the scheme, the reader is referred to [19].

Let M be the monoid $(\mathbb{F}_2^n, \cdot)$, with the structure defined by component-wise multiplication. The scheme is defined as follows:

- `Setup`(1^λ): Choose the dimension parameter $n = n(\lambda)$, and the integers $d = d(\lambda)$, $s = s(\lambda)$ such that $n = 2sd$.
- `SecretKeygen`: Choose a subset S of $\{1, 2, ..., n\}$ of size s. Set the secret key to be S.
- `E`: To encrypt a bit $m \in \mathbf{F}_2$, choose d random numbers $i_1, i_2, ..., i_d$ from the set $\{1, 2, ..., n\}$ such that there are exactly m of them in the secret key set S. Set $E(m)$ to be the vector in M, whose components corresponding to the indices $i_1, i_2, ..., i_d$ are equal to 0 and the others are equal to 1.
- `D`: To decrypt a cyphertext c using the secret key S, set $D(c) = 0$ if c has at least one component equal to 0 corresponding to a index from S and $D(c) = 1$, otherwise.

It is an easy exercise to check that $D(c_1 \cdot c_2) = D(c_1) \cdot D(c_2)$, so that the above scheme is a compact monoid homomorphic encryption scheme. This scheme proves to be efficient for the encryption and decryption times as well as for computing the AND-gate on encrypted data. Please notice that in order to compute the erosion, we only need the computation of the AND-gates on the encryption of the binary picture. Therefore, the use of the above scheme seems the most appropriate.

We shall now present a ring homomorphic encryption scheme that uses the above monoid homomorphic encryption scheme. It was presented in [16] (see also[18]).

Let $R = \mathbb{F}_2[M]$ be the monoid algebra over $\mathbb{F}_2$ generated by the monoid M presented in the previous section. The scheme is as follows:

- `Setup`(1^λ): Set-up the parameters for $(M, \mathbb{F}_2, E, D)$ monoid encryption scheme.
- `SecretKeygen`: Choose the secret key for $(M, \mathbb{F}_2, E, D)$.
- `Enc`: To encrypt a bit $m \in \mathbf{F}_2$, set $\mathtt{Enc}(m) := \mathtt{E}(m)$.
- `Dec`: To decrypt a cyphertext $c = \sum[x] \in R$ put $\mathtt{Dec}(c) := \sum \mathtt{D}(x)$.

Notice that since `D` is a homomorphism of monoids, then we get that $\mathtt{Dec}(c_1 \cdot c_2) = \mathtt{Dec}(c_1) \cdot \mathtt{Dec}(c_2)$ and since we work in the monoid algebra R, we have also that $\mathtt{Dec}(c_1 + c_2) = \mathtt{Dec}(c_1) + \mathtt{Dec}(c_2)$.

Therefore, one can compute homomorphically any circuit with polynomial size addition depth on encrypted data.

Remark: Maybe it is useful to think about an element $c = \sum[x] \in \mathbb{F}_2[M]$ as a set of elements in M. Then the addition of two ciphertexts correspond to the disjunctive union (symmetric difference) of the two sets.

IV. Experiments

We used the above mentioned scheme to compute morphological primitives over an encrypted image, and also we compare our results with other LWE based schemes such as BGV (HElib) and BFV (Microsoft SEAL).

As input we got a binary image which will be encrypted in $\mathbb{F}_2$ space and we plan to obtain the opening of this image without decrypting any intermediary results. In order to do this we must first compute the erosion of image followed by the dilation. Also, it must be said that in order to compute the closing we can encrypt the complementary image without further computations over the encrypted data and the dilation can be computed as the erosion of complementary image. To be more specific, both operations: erosion and dilation are the same as computing complexity, the only difference lays in the initial encoding of image.

This experiments were carried on a virtual machine having an Intel CPU (I7-4770, 4 cores, 3.4GHz, 12GB RAM) using only on CPU core. The Table I presents an overview of processsing times based on the hamming weight of the structural element. All the values are computed for 128 bits of security with the scheme presented in chapter 3.

The erosion time increases linear with the hamming weight because of the number of multiplications involved by the structural elements, this can be seen in Figure 6. As for the opening time, this increases exponential because of the ciphertext size which grows exponential by each operation (after

TABLE I
PROCESSING TIME FOR MORPHOLOGICAL PRIMITIVES

Hamming weight	3	5	7
Erosion	12.82 μs	24.71 μs	32.77 μs
Opening	217.88 μs	649.35 μs	2420.72 μs

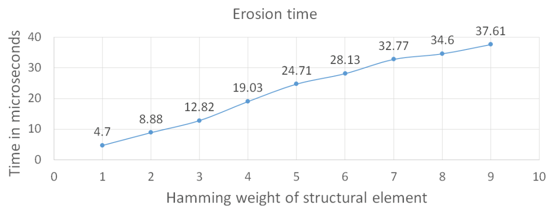

Fig. 6. Erosion times

an initial addition was carried to obtain the complementary image), this can be observed in Figure 7.

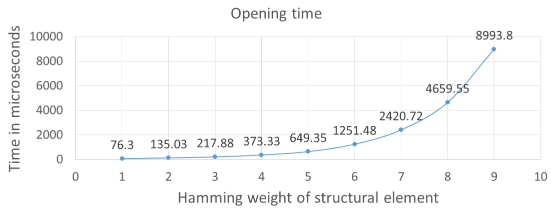

Fig. 7. Opening times

We compared our results with other schemes to see how different approaches of FHE schemes adapt to this specific operations. The results can be seen in Table II. All measured values are presented for a 128 bits of security setup using the cross structural element which has a hamming weight of 5. During the conducted experiments, all three implementations did not make use of Single Instruction, Multiple Data (SIMD) capabilities present in the LWE schemes or CSGNHE scheme, the latter being also capable to perform SIMD operations.

TABLE II
TIME COMPARISON (PER PIXEL)

Scheme	**Encyption**	**Erosion**	**Opening**	**Decryption**
CSGN	55.02 μs	24.71 μs	649.35 μs	6.21 μs
BGV	8.08 ms	262.33 ms	528.9 ms	12.54 ms
BFV	5.66 ms	99.5 ms	199.34 ms	1.19 ms

The only difference lays down in the maximum depth circuit which can be evaluated: 12 in BFV, 10 in BGV and unlimited in CSGNHE since there is no error involved. In Figure 8 we have tested how far can we go with the hamming weight until LWE schemes become faster than CSGNHE.

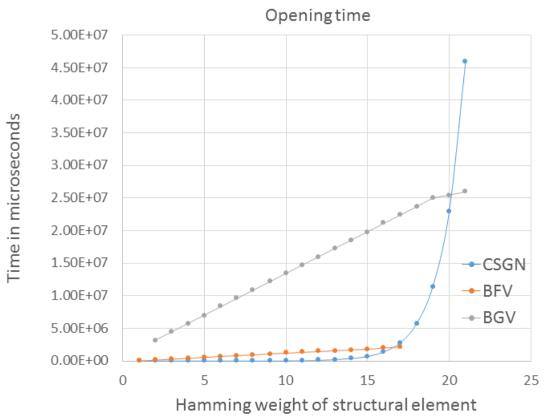

Fig. 8. Comparison of opening times

Regarding the behavior of LWE schemes (BGV and BFV) timings, we can argue that erosion and dilation primitives have a linear increase by the hamming weight of the structural element, until it will reach the noise budget. This can be seen in Figure 9.

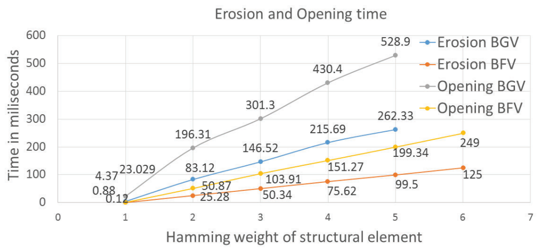

Fig. 9. Erosion and Opening times

Metrics such as fresh encrypted image size or encrypted image size after opening can be analyzed in Table III. Memory usage in a real use case scenario is directly proportional to the size of the processed image.

TABLE III
MEMORY COMPARISON (PER PIXEL)

Scheme	**Fresh ciphertext size**	**Opening ciphertext size**
CSGN	9.96 KB	297.015 KB
BGV	476.47 KB	852.75 KB
BFV	548.93 KB	548.93 KB

V. CONCLUSION

In this paper we presented a privacy oriented approach of morphological operations over digital images by using homomorphic encryption schemes.

In the literature there are multiple encryption schemes which can run computation over encrypted data, most of them being based on learning with error computational problem. We implemented two basic morphological primitives (erosion and dilation) and a third operation based on composition of the first two (opening and closing) in three different setups: the CSGNHE scheme in comparison with the well known BGV and BFV homomorphic encryption schemes.

As our experimental results state, using the cross structural element with a hamming weight of 5, there are significant advantages both in time and memory usage. We observe optimizations of 300x up to 10000x in execution times of morphological primitives and in 2.87x to 55x in memory optimization over the LWE schemes.

We conducted several tests for other structural elements with different hamming weights which hold these advantages. However, in practice there are rarely encountered situations where the structural element is bigger than 5x5 pixels (with a maximum of 25 hamming weight). The error free homomorphic encryption scheme has exponential growth in opening/closing operations, which leads to higher executions times depending on the hamming weight of the structural element: 17 in the case of BFV and 21 in the case of BGV.

As future work, we intend to make use of SIMD capabilities for the CSGNHE scheme and also to hardware accelerate the bitwise operations by using intrisic functions (e.g. Intel SSE, AVX, AVX-512 etc.) supported by the modern central processing units.

VI. Acknowledgments

This research was partially supported by the Romanian National Authority for Scientific Research (CNCS-UEFISCDI) under the project ”Advanced epidemic management techniques in the community” (PN-III-P2-2.1-SOL-2020-0148).

References

[1] Sukhvinder Singh and Surender Grewal. Role of mathematical morphology in digital image processing: A review. *International Journal of Scientific Engineering and Research*, 2:1–3, 04 2014.

[2] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, Berlin, Heidelberg, 2 edition, 2003.

[3] Shai Halevi. Homomorphic encryption. *Tutorials on the Foundations of Cryptography*, pages 219–276, 2017.

[4] Craig Gentry. *A fully homomorphic encryption scheme*. 2009.

[5] HElib, https://github.com/homenc/helib.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011.

[7] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Report 2016/421, 2016.

[8] libScarab, https://github.com/hcrypt-project/libscarab.

[9] N.P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. Cryptology ePrint Archive, Report 2009/571, 2009.

[10] FHEW, https://github.com/lducas/fhew.

[11] Leo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. Cryptology ePrint Archive, Report 2014/816, 2014.

[12] TFHE, https://tfhe.github.io/tfhe/.

[13] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachne. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. Cryptology ePrint Archive, Report 2016/870, 2016.

[14] Microsoft SEAL, https://www.microsoft.com/en-us/research/project/microsoft-seal/.

[15] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[16] Mugurel Barcu and Vicentiu Pasol. Fully homomorphic encryption from monoid algebras, US10454668B2, Oct. 2019.

[17] Vassilios Mardiris and Vassilios Chatzis. A configurable design for morphological erosion and dilation operations in image processing using quantum-dot cellular automata. *Journal of Engineering Science and Technology Review*, 9:25–30, 05 2016.

[18] Mugurel Barcau and Vicentiu Pasol. Bounded fully homomorphic encryption from monoid algebras. *IACR Cryptology ePrint Archive*, 2018:584, 2018.

[19] Mugurel Barcau, Vicentiu Pasol, and Cezar Plesca. Monoidal encryption over F2. In *International Conference on Information Technology and Communications Security - SecITC*, 2018.