Name: Luke Thomas
Project: Homomorphic Encryption Web-Based User Interface

Abstract

As cloud computing becomes more and more popular, ensuring the security of it becomes increasingly essential. Homomorphic encryption provides an efficient, secure way to perform operations on data without the need to decrypt and jeopardize its integrity. This paper is a compilation of my own work involving the development of a web-based user interface that portrays how different data types can be encrypted and decrypted using homomorphic operations on shares created from the data. For the creation of shares, Shamir's Secret Sharing Scheme was utilized because of its homomorphic properties. In this paper, the programs written to encrypt each data type in this manner are summarized. Java was the chosen language to write the programs due to its ability to easily create graphical interfaces for practical usability. Furthermore, each program with its necessary directories (if any) are packed into Java Archive files and/or ZIP files for faster download times.

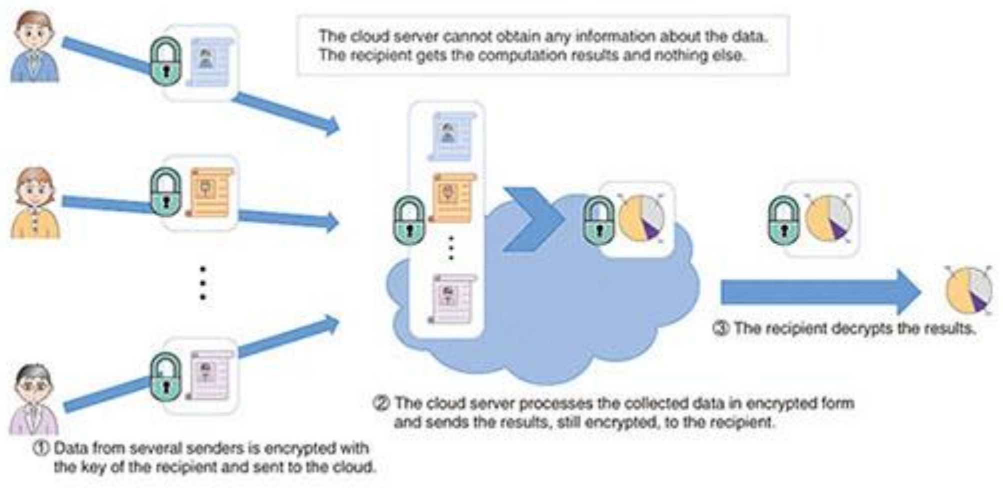
Overview and Background

Homomorphic encryption can be defined as "the conversion of data into ciphertext that can be analyzed and worked with as if it were still in its original form" (Rouse). This project is a web-based user interface in which several options for showing how homomorphic encryption manipulates data are available to use. Options for homomorphic encryption are numbers, text, and images. Each data entry is used to create multiple shares using Shamir's Secret Sharing Scheme. Operations are then performed on these shares rather than the original data.

This project is an excellent introduction to homomorphic encryption. Homomorphic encryption is the modern way of encrypting data in cloud storage to better protect information than the traditional way. As stated in the scientific paper *Homomorphic Encryption for Data Security in Cloud Computing*:

> *Traditional standard encryption methods provide security to data in storage state and transmission state. But in processing state, performing operations on data require decryption of data. At this state data is available to cloud provider. Hence traditional encryption methods are not sufficient to secure data completely.* (Chauhan)

The following diagram portrays an example of performing homomorphic encryption and processing on cloud data.


The cloud server cannot obtain any information about the data.
The recipient gets the computation results and nothing else.
③ The recipient decrypts the results.
② The cloud server processes the collected data in encrypted form and sends the results, still encrypted, to the recipient.
① Data from several senders is encrypted with the key of the recipient and sent to the cloud.

(Tibouchi)

In 1979, Shamir proposed a scheme that allowed a secret to be divided into several shares and distributed to multiple individuals (Benaloh). Shamir's (k,n)-threshold scheme is one in which the secret is divided into n shares and can be reconstructed using any k or more shares (Lin, H.). Shamir's Secret Sharing is an important tool because "in certain application cases, it is a risk if a set of secret data is held by only one person without extra copies because the secret data set may be lost incidentally or modified intentionally…(or) it might be necessary for a group of persons to share a certain set of secret data" (Lin, C.). One such scenario might be ensuring the confidentiality of digitized medical records. As stated in *Medical Image Security and EPR Hiding Using Shamir's Secret Sharing Scheme*, "Medical applications such as telediagnosis require information exchange over insecure networks" (Ulutas). This means that attackers may attempt to steal the sensitive data while it is in transit. This paper suggests using Shamir's Scheme to mitigate this risk.

Motivation

This website will serve as an interactive introduction into the homomorphic properties of Shamir's Secret Sharing for future students studying cryptographic strategies. As stated in one paper studying the impact of interactive websites on learning, "Using features which promote…interactive learning activities encourages a deep approach to learning…and enhanced understanding of content" (Kember). Being given the opportunity to provide this type of learning tool that may help others understand secret sharing and homomorphic encryption was one of the driving factors for pursuing this project.

Scope

The scope of this project includes three data types: numbers, text, and images. Programs for encryption of these data types utilize Shamir's Secret Sharing Scheme to portray its homomorphic properties. There exist two options for number encryption, four options for text encryption, and two options for image encryption. Options for number encryption include homomorphic number addition and subtraction. Options for text encryption include homomorphic text addition, subtraction, concatenation, and searching. Options for image encryption include homomorphic pixel addition and subtraction.

Limitations
To run the demo programs provided in the website, users must first download the necessary Java Archives and/or ZIP files. Demos that require the use of external files (i.e. text search, pixel subtraction, and pixel addition) are packages into ZIP files that contain the needed directories for saving and reading in text files or images. In these cases, the user must download and extract the contents of the compressed file into the same parent directory for the program to function correctly. Otherwise, the user is only required to download a JAR file.

In some cases, browsers and/or user computers will warn of potential security risks or even block the downloading of the site's files. The user will have to either confirm that they are aware of the risks or add an exception to download the demo files.

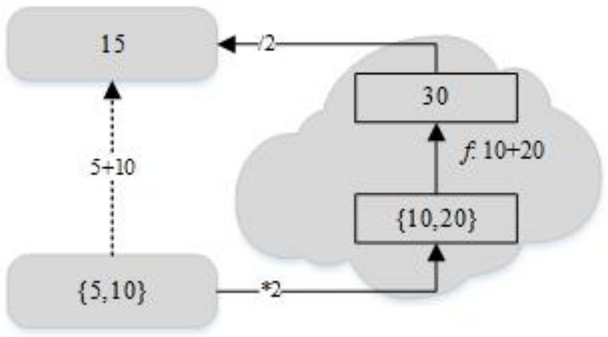
Implementation

This section is segmented into descriptions of the homomorphic properties possessed by each data type used. Each of these sections are then separated further into detailed descriptions of each homomorphic encryption technique included in each of the corresponding data type webpages.

When a user runs one of the Java Archives, they will be presented with a GUI in which they can provide their own data to see the selected homomorphic encryption techniques performed on it. If the user is required to choose one or more files, they will be provided with a file explorer to search their computer for the desired file(s). Otherwise, inputs can be typed directly into the interface.

Numbers
Numbers are just one data type that can be used in homomorphic encryption, where operations are performed on the encrypted versions of numbers rather than the original numbers. The figure below is an example of homomorphic encryption on two numbers, five and ten, whose sum can be resolved from the sum of their encrypted versions.

*This is a naïve implementation of what is included in the final product for homomorphic encryption of numbers, which includes Shamir's Secret Sharing.

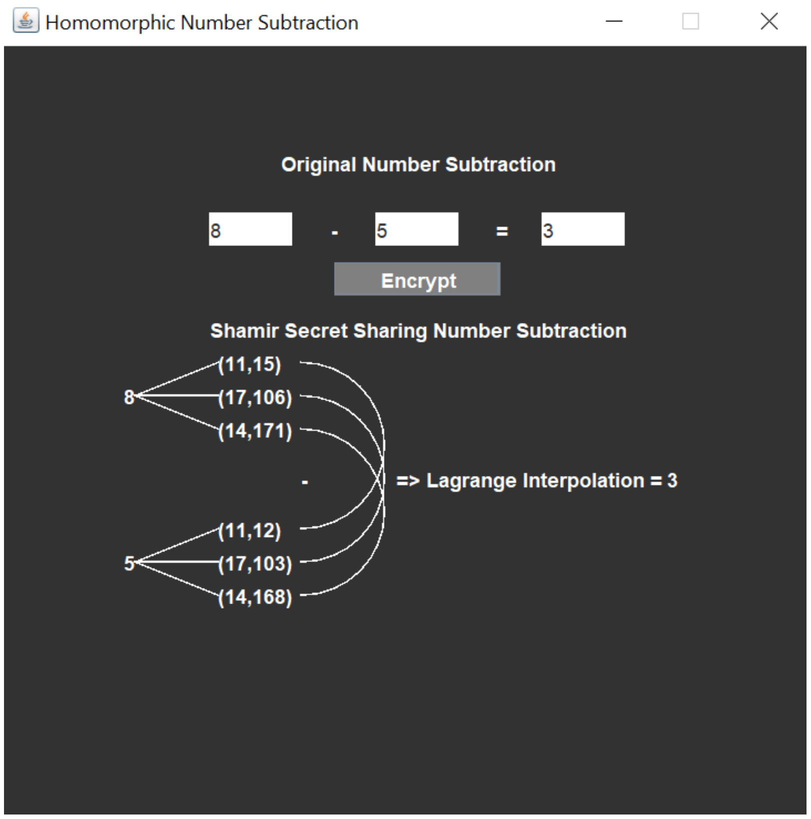


(*Fully Homomorphic Encryption Schemes*)

For the number encryption webpage, the user can choose between homomorphic number addition and homomorphic number subtraction. After downloading and running on of the demo programs, the user enters two numbers in the provided form of an equation:

A +/- B = C

Where A and B are user inputs and C is the sum of those inputs, which is automatically generated after the user clicks the "Encrypt" button. Each user-provided number is then used to create the corresponding encrypted shares. The shares of the second number are then added to or subtracted from the shares of the first number to produce the encrypted results. These results are then passed through the reconstruction function, which utilizes Lagrange Interpolation, to produce the original sum or difference for the user inputs. Below is a screenshot of the homomorphic number subtraction demo.

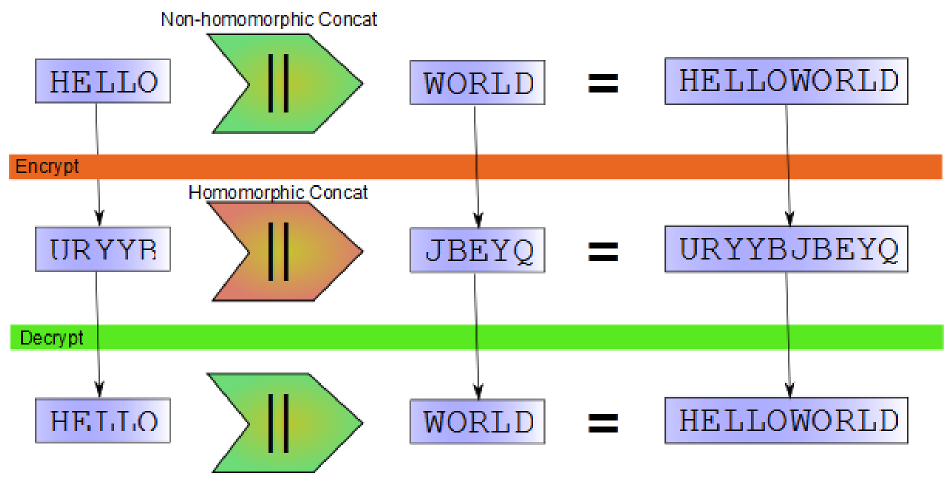

Homomorphic Number Subtraction
Original Number Subtraction
8
-
5
=
3
Encrypt
Shamir Secret Sharing Number Subtraction
8
(11,15)
(17,106)
(14,171)
-
=> Lagrange Interpolation = 3
5
(11,12)
(17,103)
(14,168)

In this scenario, the user inputs would be eight and five. The original difference is three which can be seen in the rightmost text-insert box. Below the encryption button, the shares that were generated from each user input value can be seen in the format:

(x-value, y-value)
…where x=the value used to encrypt the number and y=the encrypted value.

The curved lines connecting the shares from the eight and the five represent the subtraction operation between those shares. Finally, through Lagrange Interpolation, these share differences are used to reconstruct the desired output of three, which matches the original outcome.

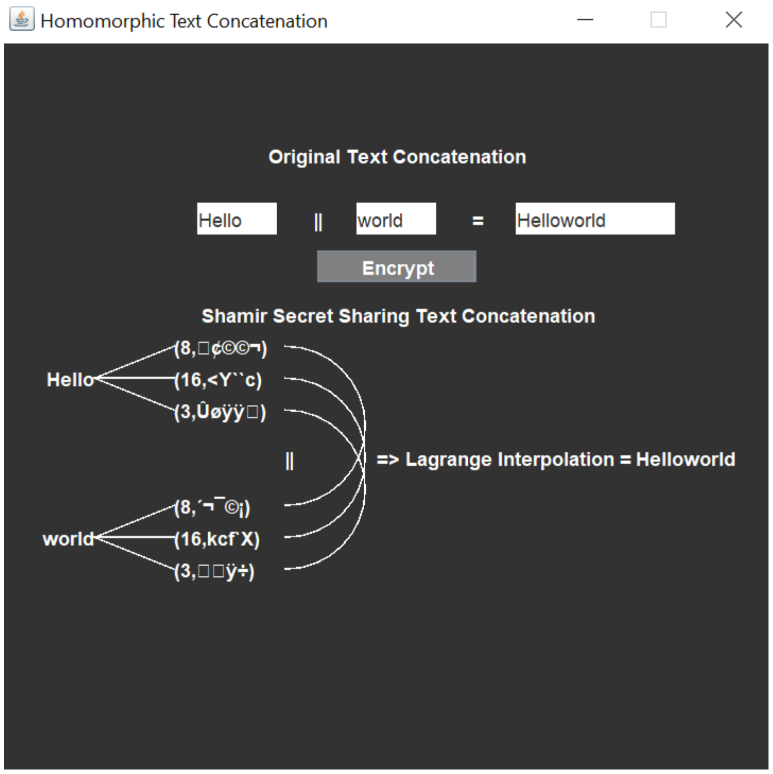
Text
Text is another data type that can be used in homomorphic operations. One example of homomorphic text encryption is concatenation, which is portrayed below.



(Stuntz)

*This is a naïve implementation of what is included in the final product for homomorphic encryption of text, which includes Shamir's Secret Sharing.

This operation is included in the text encryption webpage in addition to text addition, subtraction, and search. In homomorphic concatenation, the user provides two strings which are each used to generate corresponding shares. These shares are concatenated and the resulting strings are passed through the reconstruction function, utilizing Lagrange Interpolation, to produce a string that is identical to the original string concatenation. Below is a screenshot from the homomorphic text concatenation demo.


Homomorphic Text Concatenation
Original Text Concatenation
||
=
Encrypt
Shamir Secret Sharing Text Concatenation
||

```
Hello || world = Helloworld

Hello  (8,□¢©©¬)
       (16,<Y``c)
       (3,Ûøÿÿ□)
                     ||        => Lagrange Interpolation = Helloworld
world  (8,´¬¯©¡)
       (16,kcf`X)
       (3,□□ÿ÷)
```

In this scenario, the user provided the strings “Hello” and “world” which concatenate to “Helloworld”. The shares generated from each string are shown in the same fashion as the number subtraction demo. Users can verify that the program worked correctly by comparing the reconstructed concatenation with the original, which should be identical.

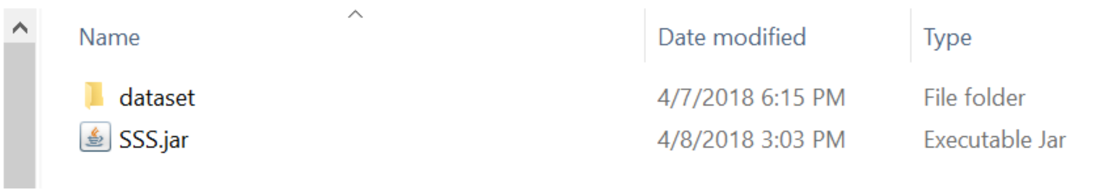
Homomorphic text addition and subtraction act very similarly to homomorphic number addition and subtraction, with the exception that the text is read in and encrypted one character at a time to allow the program to operate on the ASCII number values.

Homomorphic text search reads in a user-supplied text file and a user-supplied string to search for in the file. Shares are created for both the text file and the string. Each string share is searched for in the corresponding (same x-value) file share. If the specified string is found in the file, it is replaced by ones so the user can see all found locations more easily. The resulting files from the text searches are then used to reconstruct the original text file with all found string matches replaced with ones. The user can view each file by clicking the corresponding buttons provided in the graphical user interface.

Images

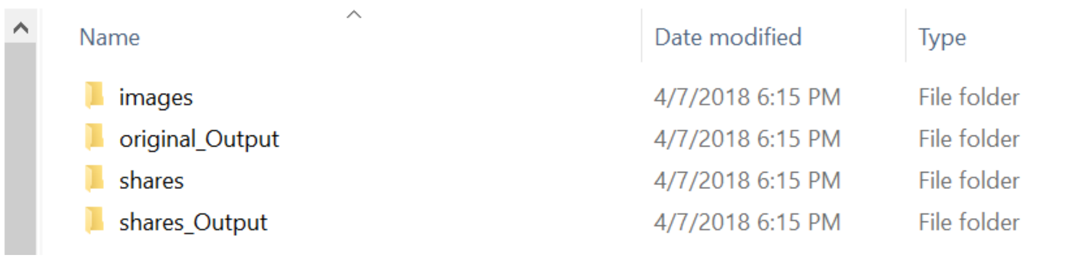
For homomorphic image encryption, the user can either choose two images from their personal storage or two sample images from the provided directory, named “images”, included in the ZIP file. Shares are created from these images on which a user-chosen homomorphic encryption method is used (addition or subtraction). The results from these share operations are then used to reconstruct the secret image, which is either the sum of the pixels from the image inputs or their difference. Users can view the images in larger formats by navigating through the provided directories. The file structure can be seen below.

Parent directory



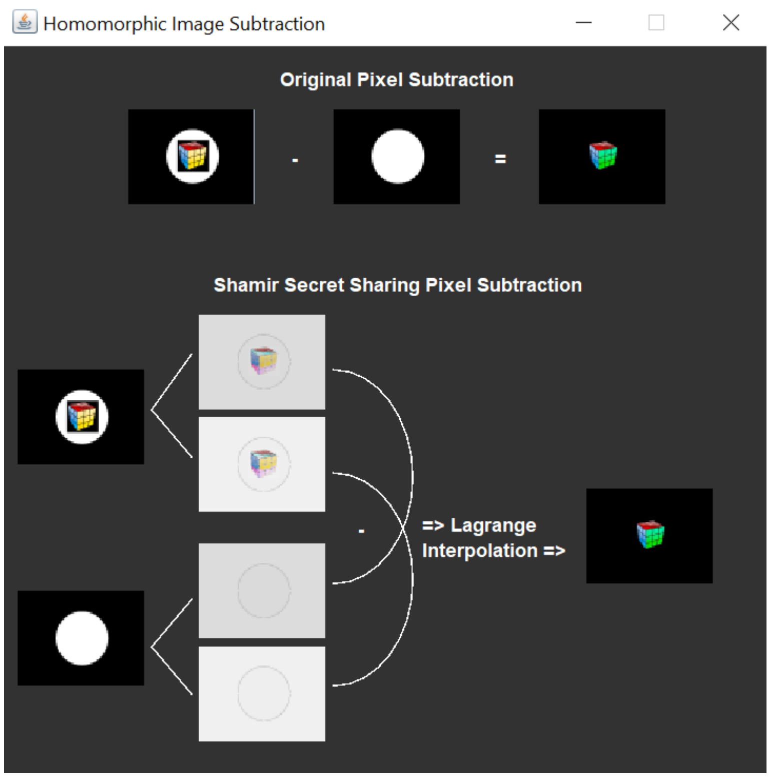
Name	Date modified	Type
dataset	4/7/2018 6:15 PM	File folder
SSS.jar	4/8/2018 3:03 PM	Executable Jar

Contents of “dataset”



Name	Date modified	Type
images	4/7/2018 6:15 PM	File folder
original_Output	4/7/2018 6:15 PM	File folder
shares	4/7/2018 6:15 PM	File folder
shares_Output	4/7/2018 6:15 PM	File folder

The directory “images” contains sample images that users can choose from, as previously stated. The directories “original_Output” and “shares_Output” will contain the original image addition/subtraction output and the reconstructed output, respectively. Lastly, “shares” will contain all image shares and share operation results. Below is a screenshot of the homomorphic image pixel subtraction demo.


Homomorphic Image Subtraction
Original Pixel Subtraction
-
=
Shamir Secret Sharing Pixel Subtraction
-
=> Lagrange Interpolation =>

In this scenario, the user selected the images of a circle containing a cube and of the same circle, but empty. Original pixel subtraction produced the image on the right side of the equal sign in the top portion of the window. Image shares were then generated and the subtraction operation was performed on each set. Following reconstruction, the resulting image can be seen in the bottom righthand side of the interface. If the process worked correctly, these two resulting images should be identical.

Applications Used

Application	**Usage**
Notepad++	Webpage development
IntelliJ IDEA Community Edition	Encryption programs
GitHub.com	Website hosting

References

1. Rouse, M. (n.d.). Homomorphic Encryption. Retrieved March 21, 2018, from http://searchsecurity.techtarget.com/definition/homomorphic-encryption

2. Chauhan, K., Sanger, A., & Verma, A. (n.d.). Homomorphic Encryption for Data Security in Cloud Computing. Retrieved March 28, 2018, from http://ieeexplore.ieee.org/document/7437616/?reload=true

3. Tibouchi, M. (n.d.). Fully Homomorphic Encryption over the Integers: From Theory to Practice. Retrieved April 16, 2018, from https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201407fa5.html

4. Benaloh, J. C. (1986, August 11). Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract). Retrieved April 17, 2018, from https://link.springer.com/chapter/10.1007/3-540-47721-7_19

5. Lin, H., & Harn, L. (1991, November 11). A Generalized Secret Sharing Scheme with Cheater Detection. Retrieved April 18, 2018, from https://link.springer.com/chapter/10.1007/3-540-57332-1_12

6. Lin, C., & Tsai, W. (2003, December 05). Secret Image Sharing with Steganography and Authentication. Retrieved April 16, 2018, from https://www.sciencedirect.com/science/article/pii/S0164121203002395

7. Ulutas, M., Ulutas, G., & Nabiyev, V. V. (2010, December 08). Medical image security and EPR hiding using Shamir's secret sharing scheme. Retrieved April 16, 2018, from https://www.sciencedirect.com/science/article/pii/S0164121210003274

8. Kember, D., et al. (2010, May 20). Understanding the ways in which design features of educational websites impact upon student learning outcomes in blended learning environments. Retrieved April 18, 2018, from https://www.sciencedirect.com/science/article/pii/S0360131510001478

9. Fully Homomorphic Encryption Schemes. (n.d.). Retrieved March 27, 2018, from http://cryptowiki.net/index.php?title=Fully_homomorphic_encryption_schemes

10. Stuntz, C. (2010, March 18). What is Homomorphic Encryption, and Why Should I Care? Retrieved March 27, 2018, from https://community.embarcadero.com/blogs/entry/what-is-homomorphic-encryption-and-why-should-i-care-38566