SECRET SHARING APPROACH FOR SECURING CLOUD-BASED IMAGE PROCESSING

MANORANJAN MOHANTY

A THESIS SUBMITTED FOR
THE DEGREE OF
DOCTORATE OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2013

SECRET SHARING APPROACH FOR SECURING CLOUD-BASED IMAGE PROCESSING

MANORANJAN MOHANTY

A THESIS SUBMITTED FOR
THE DEGREE OF
DOCTORATE OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

Under the Supervision of

ASSOCIATE PROFESSOR WEI TSANG OOI

2013

Declaration

I hereby declare that this thesis is my original work and that it has been written by me in its entirety. I have duly acknowledged all the sources of information consulted for the thesis.

This thesis has also not been submitted for any degree in any university previously.

Manoranjan Mohanty

DEDICATED TO MY MOTHER, BHARATI MOHANTY

Acknowledgement

First and foremost, I would like to thank my supervisor, Professor Wei Tsang Ooi for giving me the skills to think logically and practically, write efficiently, and communicate clearly. These three skills have been crucial elements in the realization of this thesis. Having embarked upon this PhD with unrefined research skill and technical communication abilities, Professor Ooi's guidance has helped me to hone my skills and complete this work in timely manner. I would also like to express my gratitude to him for sending me on two internships: one at the National Institute of Informatics in Japan and other at the University of Winnipeg in Canada.

I take this opportunity to express my profound gratitude to Professor Pradeep K. Atrey at the University of Winnipeg for being my internship advisor in Canada, and for being a key collaborate of my project. Professor Atrey has helped shape my understanding of the security and privacy issues in cloud-based systems, and has spent a great deal of time discussing different approaches to solving these issues. I would also like to extend my sincere gratitude to him and his family for the help that they extended during my stay in Canada.

I sincerely thank Professor Helmut Prendinger, my internship advisor at the National Institute of Informatics in Japan for giving me the opportunity to work on the OpenPDA project. The development skills that I acquired from my work on this project were utilized in implementing my research findings.

My sincere thanks go to Professor Mohan Kankanhalli and Professor Roger Zimmermann for their time spent evaluating my work and my thesis. Their comments have been invaluable in the development and improvement of this work.

While this thesis could not have been a reality without the assistance of my National University

of Singapore (NUS) instructors, I would never have found myself at NUS without teaching of my past educators. I would therefore like to thank all of the educators in my past who have supported me and shared with me their invaluable knowledge over the years. Teachers such as *Nayak-Sir* (Mr. Abhimanyu Nayak who taught me in my secondary school) provided me with the mentorship and encouragement to guide me on my path of learning. I would also like to extend my thanks to Professor K K Bharadwaj, Professor Sonajharia Minz, Professor R K Agarwal, Professor D P Vidyarthi, Doctor D K Lobiyal, and Mr. Sushil Kumar of JNU, and Mr. Debashish Rath for their recommendations during my PhD applications.

A thesis is not only a technical document, but also an amalgamation of the skills and lessons learned throughout one's education. The life of a PhD student is not an easy one, and I would like to extend my gratitude to those who have supported me, and those who have challenged me. The latter helped to prepare me for the challenge ahead, and the former provided me the with the support I needed to surmount the obstacles that I've encountered along the way. While naming all of the individuals who provided me with unconditional moral and/or technical support during my PhD studies is difficult, I will like to name a few. In no particular order, I would like to extend my gratitude to Atala Panda, Manoranjan Patnaik, Sushant Swain, Asnika Das, Bibekananda Mishra, Deven Balani, Asit Sahoo, Shreyas Behera, Ajay Sinha, Pushkar Kaushik, Rameshwar Pratap Yadav, Deependra Singh Chauhan, Amit Chouhan, Akash Mishra, Ranjit Rajak, Uma Shanker, Upakul Barkakaty, Priyank Singh, Anil Gupta, Vinay Bharadwaj, Shreelatha Rakesh, Shital Mishra, Sucheendr Kumar, Sudipta Chattopadhyay, Sriganesh Srihari, Wang Hui, Zhao Zhenwei, Girisha De Silva, Shant Sagar, Le Duy Khanh, Rajiv Ratn Shah, Mukesh Prasad, and Neeraj Singh Chauhan.

Finally, I would like to thank my family: my parents, siblings, cousins, uncles and aunts, and all other relatives for their support and guidance. On the same note, I would like to thank Doctor Bijay Patnaik of Sudarshan Mahavidyalaya for his guidance and genuine caring for me. Doctor Patnaik has been my role model and my mentor. I would also like to extend my sincere thanks to Doctor Bipin Senapati of Raipur Village for his moral and financial support to my studies.

Abstract

Cloud-based imaging, which is being increasingly used to store and process volume data/images, presents security and privacy challenges. Although these challenges have been addressed for cloud-based storage, to the best of our knowledge, they are still a concern for cloud-based volume data/image processing, such as image scaling/cropping and volume ray-casting. In this thesis, we address this concern for cloud-based image scaling/cropping and cloud-based volume ray-casting by using Shamir's (k, n) secret sharing and its variant (l, k, n) ramp secret sharing, which are homomorphic to addition and scalar multiplication operations, to hide volume data/images in datacenters.

Firstly, we address the incompatibility issue of the floating point operations of a volume data/image processing algorithm with the modular prime operation of Shamir's secret sharing either by converting the floating point operations to fixed point operations or by excluding the modular prime operation from secret sharing. Our analysis shows that the former technique can degrade the image quality and the latter can degrade security.

Then, we integrate secret sharing with image scaling/cropping, pre-classification volume ray-casting, and post-classification volume ray-casting, and propose three cloud-based frameworks. The frameworks have been designed with the philosophy that a server secret shares volume data/image and distributes the shares (i.e., hidden data/images) among n datacenters; a datacenter, upon request from a user, processes the hidden volume data/image, and sends the processed volume data/image (which is also hidden) to the user; and the user recovers the secret processed volume data/image from k hidden processed volume data/images. Experiments and analyses show that our frameworks can provide data confidentiality, data integrity, and data availability; and can incur low computation cost to the user.

Contents

List of Figures

List of Symbols

Symbol	**Meaning**
a_i	Polynomial in the secret sharing function
b	Number of bits required to represent a pixel
c	Total number of sample points
d	Rounding bits
f	Rounding bits
g	Rounding bits
k	Minimum number of shares required to construct a secret
l	Number of secrets in ramp secret sharing
n	Total number of shares created from a secret
p	Share number
q	Prime number
q'	Prime number
s	A sample point in V
s_p	A sample point in V_p
$t_i(x)$	Lagrange basis function
x_i	i^{th} share number
v	A data voxel in V
v_p	A data voxel in V_p
A_i	Opacity of i^{th} voxel or i^{th} sample point

A	Composited opacity along a ray
$C_i^{\uparrow}$	Classified color of i^{th} data voxel
C_i	Shaded color of i^{th} voxel or i^{th} sample point
C	Composited color of a pixel
C'	Scaled composited color
D_i	Interpolation factor of i^{th} voxel or i^{th} sample point
$F(x)$	Secret sharing polynomial for Shamir's secret sharing
$F'(x)$	Secret sharing polynomial for modified Shamir's secret sharing
G_i	Gradient of i^{th} data voxel or i^{th} sample point
$L(x)$	Lagrange interpolated polynomial
$L'(x)$	Lagrange interpolated polynomial without the modular prime operation
(S,T)	(x,y) coordinate of a pixel in the image space
$PID_{(S,T)}$	Proxy for the coordinate (S,T)
$N(s)$	Set of eight neighbouring voxels of sample point s
$\mathcal{N}_i$	Normal of i^{th} data voxel or i^{th} sample point
O_i	Variable used in composition
P_i	Scalar value of i^{th} data voxel or i^{th} sample point
V	Given volume data
V_i	i^{th} share of V
$X_{i,p}$	p^{th} share of a variable X_i
$X_i^{(d)}$	Fixed point representation of a variable X_i obtained by first rounding off X_i by d decimal places and then multiplying 10^d by the rounded off value
X'_s	Scaled interpolated value of X for a sample point s
Y_i	Addition of ambient reflection coefficient and diffuse reflection coefficient of i^{th} voxel or i^{th} sample point

Z_i	Specular coefficient of i^{th} voxel or i^{th} sample point
$\epsilon_{x,y}$	Roundoff error due to rounding off x by y decimal places
ϵ_{X_s}	Error in interpolation of X value for the sample point s
$\epsilon_{C_s^{\uparrow}}$	Error in the classified color
ϵ_{A_s}	Error in the classified opacity
ϵ_C	Error in color composition
ϵ_A	Error in opacity composition
$\epsilon_{C,max}$	Upper bound of ϵ_C
$\epsilon_{C,min}$	Lower bound of ϵ_C
$\epsilon_{C,eff}$	Difference in the color value of a pixel due to roundoff error ϵ_C
α_p	Secret sharing variable for V_p

Chapter 1

Introduction

1.1 Motivation

Digital imaging is being increasingly used in a variety areas of daily life such as medicine [1, 2], personal photography, teaching and learning [3], etc. In this technique, a pipeline of five main steps (shown in Figure 1.1): *data capturing*, *data preprocessing* (which removes unnecessary details, such as noise, from captured data), *data-to-image conversion*, *image processing* (such as scaling and cropping), and *image display*, are followed to produce an image from an object, and display the image to a user.

With advances in telecommunication, remote digital imaging techniques, such as teleradiology or telepathology, have become popular. In these techniques, the intermediatory imaging steps, such as data preprocessing, data-to-image conversion, and image processing, are performed by the server.

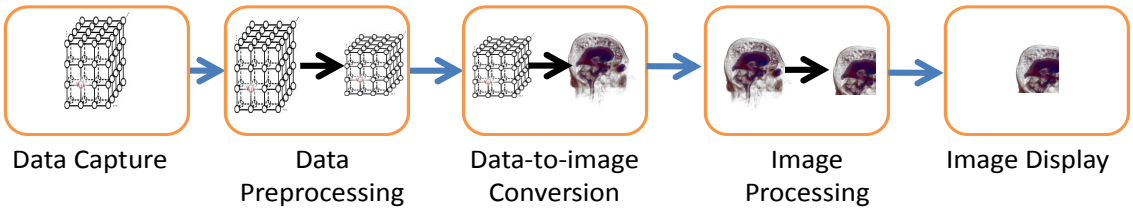


Figure 1.1: Digital imaging pipeline.

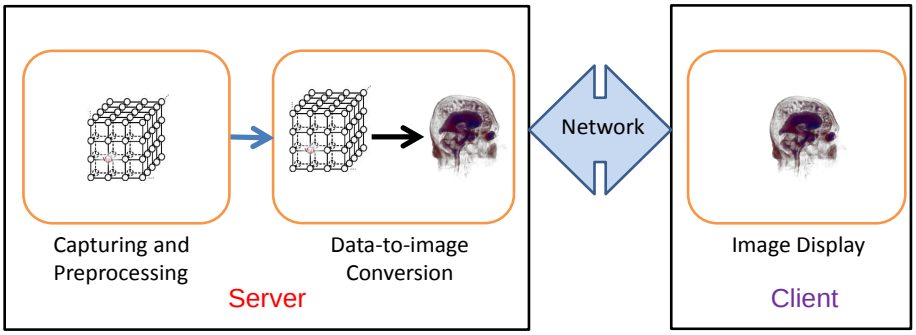


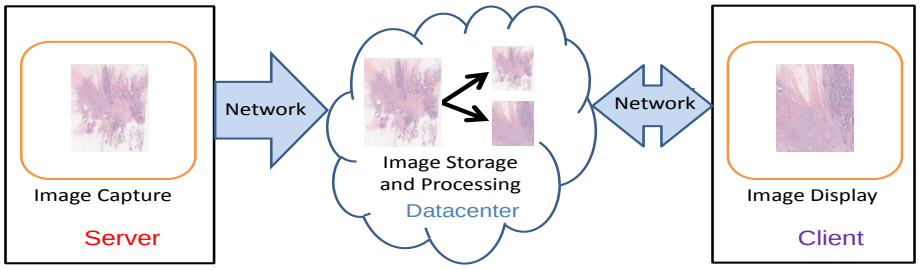
Figure 1.2: Server-side rendering.

For example, in the case of 2D image visualization, a server typically converts data to an image, as the required operations are implicit in a data capturing device. Similarly, in the case of 3D image visualization, server-side data rendering (shown in Figure 1.2), which both captures and renders data at the server end, is often used [4, 5, 6]. Furthermore, in an image streaming framework, image processing is also performed at the server end.

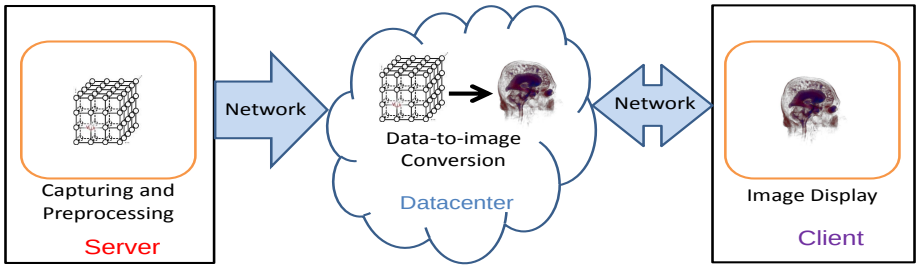
With the increase in the size of an image and the requirement of managing multiple users, it is no longer feasible for an organization, such as a hospital, to store and process large data/images. For example, storing and processing huge *whole slide images*, each having a size in the scale of tens of GBs in uncompressed form [7], presents a scalability issue. Therefore, organizations are relying on third party cloud datacenters for the storage, processing, sharing, and management of data/images. In addition to being more scalable, such cloud-based imaging solutions are more economical, offer better computing resources, and can produce lower visualization latency by storing/processing the data/image in a datacenter closer to the user.

Three important data/image processing schemes are image scaling/cropping for 2D image visualization, and pre-classification volume ray-casting and post-classification volume ray-casting for 3D image visualization. Downloading a large image, such as a whole slide image to users may not be always feasible. Users may want to preview a scaled down version of the image before deciding whether to download the image. Further, users may just want view a particular region of interest in the image, in which case, a cropped region should be downloaded. These two operations, scaling and cropping, can be combined to support zooming and panning, two natural user interactions to

The voxel grid figure has been obtained from http://johnrichie.com/V2/richie/isosurface/volume.html



(a) 2D image visualization



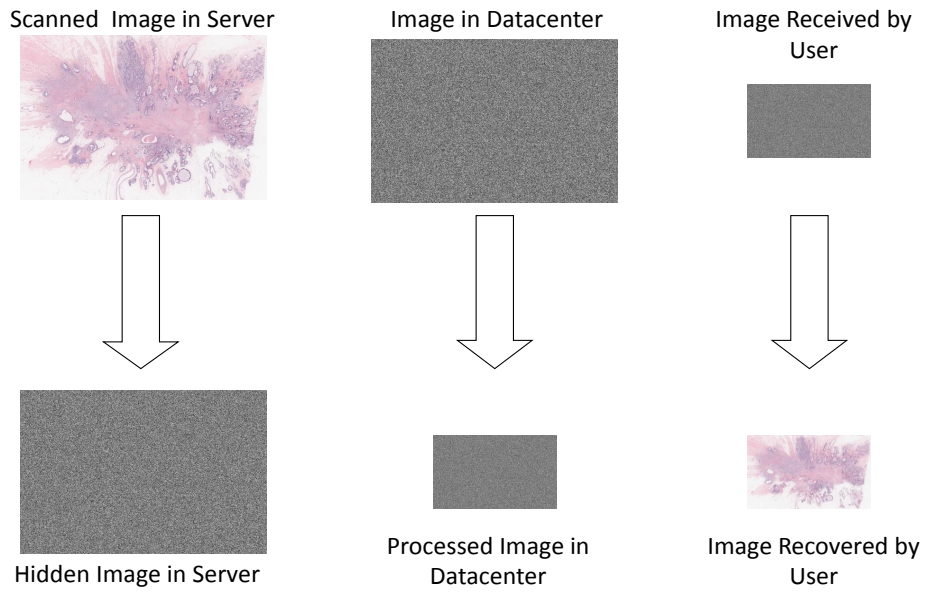
(b) 3D image visualization

Figure 1.3: Cloud-based image visualization

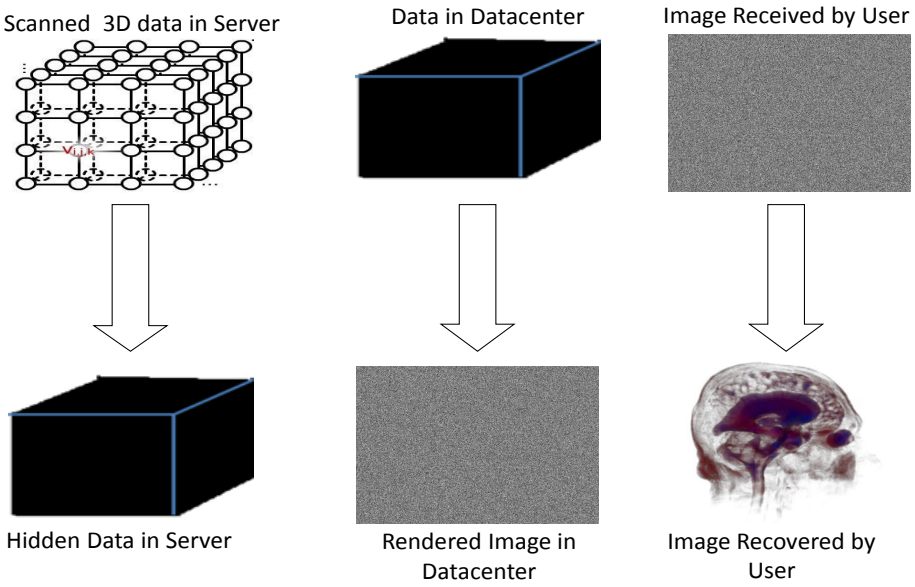
explore large images. On the other hand, the volume rendering schemes produce an image from the physical properties of an object. Among the volume rendering schemes, volume ray-casting algorithms are popular since they render better quality image than other rendering schemes [8].

Figure 1.3 shows the cloud-based image scaling/cropping and cloud-based volume ray-casting scenarios. As shown in the figure, in the case of 2D image visualization, a datacenter scales/crops an image [9]), and in the case of 3D image visualization, a datacenter renders a 3D image from a 3D volumetric data [10, 11, 12].

Although third-party cloud-based volume data/image storage and processing has many advantages, the security of the volume data/image and the privacy of the owner of the volume data/image are two main concerns [13, 14]. For example, in the case of medical imaging, an adversary can access a datacenter that stores medical volume data/images of patients and misuse the information in several ways. Firstly, for economical benefits, the adversary may illegally sell the disease information of patients to other interested parties such as insurance companies (*confidentiality issue*). Secondly, a medical image can be tampered to provide misleading information to doctors (*integrity issue*). Thirdly, for publicity, both the health information and the name of the admitting hospital of a prominent person can be disclosed to unauthorized individuals (*privacy issue*). Due to these poten-



(a) Secure cloud-based 2D image visualization



(b) Secure cloud-based 3D image visualization

Figure 1.4: Our objective for 2D/3D image visualization

tial threats, laws, such as the *HIPPA* act in USA, the *PIPED* act in Canada, and the *Data Protection Act* in European countries, have been enacted to protect the data/images of citizens.

A common approach to addressing the security and privacy issues is the use of cryptographic techniques to hide important information of volume data/images from the datacenters. Although this solution is available for cloud-based volume data/image archives [15, 16], such a solution does not exist for cloud-based image scaling/cropping or cloud-based volume ray-casting.

1.2 Problem Statement

This thesis focuses on performing image scaling/cropping and volume ray-casting operations in hidden domain.

We assume that (i) the server, which owns the secret data/image, and outsources storage/processing to n datacenters is secured (no adversary can access the server); (ii) a datacenter is honest (honestly performs requested operations), but can be curious (can try to know content of data/image); (iii) no more than $k-1$ (where, $k \leq n$) datacenters can communicate with each other; and (iv) the client is secured. Furthermore, we also assume that an adversary cannot access the communication links, storage devices, or processing devices of k or more datacenters at any point of time.

Our objective is to hide the volume data/image S from a datacenter using a cryptosystem $H(\cdot)$, and allow operation $R(\cdot)$ on the hidden volume data/image $H(S)$ such that: (i) the datacenter cannot know application-specific confidential information of the secret volume data/image S or the secret processed volume data/image $R(S)$ from the hidden volume data/image $H(S)$ or the hidden processed volume data/image $R(H(S))$, (ii) a client can recover $R(S)$ from at least k $R(H(S))$'s, (iii) a client can detect tampering on $H(S)$ or $R(H(S))$ when $n \geq k$, and (iv) a client will able to recover a $R(S)$ even if $n-k$ datacenters cannot participate. We mainly focus on three commonly used volume data/image processing schemes: image scaling/cropping, pre-classification volume ray-casting algorithm, and post-classification volume ray-casting. We illustrate the objective in Figure 1.4, which shows that: (i) the server hides volume data/image before sending it to the datacenters, (ii) a datacenter renders hidden volume data to produce noise-like rendered image, or scales/crops a hidden image to produce a noise-like scaled/cropped image, and (iii) the user recovers the secret

image from the noise-like rendered or scaled/cropped images.

Note that our approach can introduce overhead as it performs extra operations by requiring a server to hide the volume data/image, and a client to recover the hidden image. One of our objectives is to lessen this overhead. Furthermore, we also want to provide information theoretical security to protect the volume data/image from an adversary having unlimited computational capability. In summary, we aim to find solutions keeping the following points in mind.

(i) **Confidentiality:** Neither the original volume data/image nor the processed volume data/image should disclose any information to a datacenter.

(ii) **Integrity:** Tampering with volume data or images in a datacenter should be detected by the user.

(iii) **Availability:** A user should be able to obtain the requested volume data/images even if some datacenters are unable to function.

(iv) **Privacy:** The privacy of a person associated with the volume data/image (for example, a patient in the case of medical imaging) should be preserved.

(v) **Computational efficiency:** The computational overhead in processing volume data/images should be minimized. The computation cost in recovering secret volume data/images should be suitable for visualization latency, and the computations should be supported by the user's computing device.

(vi) **Bandwidth efficiency:** The data overhead in transmitting hidden volume data/images from a datacenter to a user should be minimal and suitable to the Web.

(vii) **High quality image:** Any degradation in image quality should be minimal.

Fulfilling all the above requirements may be difficult as it is very likely for there to be a tradeoff between them. For example, it may be difficult to provide both high security and low overhead together. One of our goals is to study the tradeoffs carefully and propose application-specific solutions.

1.3 Technical Challenges

Volume data/images can be hidden from a datacenter by applying a cryptosystem, such as a data encryption technique [16] or secret sharing [17], at the server end. The applied cryptosystem, however, should be homomorphic to the mathematical operations performed on the volume data/images to ensure that the required secret image can be recovered from the processed hidden volume data/images. In other words, if the cryptosystem hides a secret volume data/image S with an operation $H(\cdot)$, a datacenter performs the $R(\cdot)$ operation on the hidden volume data/image $H(S)$, and a user recovers the processed secret image from the processed hidden volume data/image $R(H(S))$ with an operation $H^{-1}(\cdot)$, then the condition

$$R(S) \approx H^{-1}(R(H(S)))$$

must hold.

Finding a cryptosystem that is homomorphic to the operations performed by a datacenter is a concern. Available fully homomorphic cryptosystems have impractical overheads [18, 19], and available somewhat homomorphic cryptosystems are only homomorphic to certain imaging operations [20].

Furthermore, any selected cryptosystem operates on a finite field GF(q) [21], and is therefore incompatible with the floating point operations of a volume data/image processing algorithm. For example, even if Shamir's secret sharing is homomorphic to addition and scalar multiplication [22], we cannot use it to secure the polynomial interpolation used in most imaging techniques, since it requires modular prime operations. This incompatibility issue is another concern.

1.4 Summary of Contribution

Our work first addresses the technical challenges discussed above, and then applies the solutions to design a cloud-based image scaling/cropping framework and two cloud-based volume rendering frameworks. These frameworks can hide application-specific confidential information from a datacenter, detect tampering on volume data/image, and operate even when certain number of

datacenters do not participate. For simplicity, we call our image scaling/cropping framework as *secure cloud-based image scaling/cropping framework*, the volume rendering framework using pre-classification volume ray-casting as *secure cloud-based pre-classification volume ray-casting framework*, and the volume rendering framework using post-classification volume ray-casting as *secure cloud-based post-classification volume ray-casting framework* in the rest of this thesis.

1.4.1 Choosing a cryptosystem

Since one of our objectives is to lessen the overheads of our frameworks, we choose to use a somewhat homomorphic cryptosystem over a fully homomorphic cryptosystem. There are two main types of somewhat homomorphic cryptosystems: (i) secret sharing-based schemes, and (ii) public key encryption-based schemes. The encryption-based schemes assume that an adversary does not posses enough computational power to decrypt a publicly available encrypted message in reasonable time period. Therefore, these schemes are conditionally secured as it can be possible for the adversary to get required computation power. The secret sharing based schemes, on the other hand, do not assume about the computational power of the adversary. These schemes hide a secret by not disclosing enough information about the secret to the adversary. Therefore, secret sharing schemes provide better confidentiality than encryption-based schemes. Secret sharing schemes are typically used to store highly important information such as cryptographic keys [23] [24], military data [25] etc. Furthermore, without using an additional trick, secret sharing schemes can simultaneously provide data confidentiality, data integrity, and data availability. Therefore, we short-list secret sharing schemes for our framework. Among the secret sharing schemes, Shamir's secret sharing is more efficient than other secret sharing schemes such as Blakley's secret sharing and Chinese Reminder Theorem-based secret sharing schemes. Therefore, we choose Shamir's secret sharing for our work.

1.4.2 Addressing incompatibility of Shamir's secret sharing with modular prime operation

We address the incompatibility issue of Shamir's secret sharing with a volume data/image processing algorithm in two ways. First, similar to the parallel work of Finamore [26], we exclude the modular prime operation from Shamir's secret sharing [27]. This approach, however, can degrade

security as the modified secret sharing no longer works in GF(q). Second, based on Catrina et al.'s proposal [28], we modify the floating point operations of a volume data/image processing algorithm to fixed point operations by modifying a floating point number to a fixed point number. This technique, however, rounds off a floating point number, and therefore introduces round-off error.

1.4.3 Proposed frameworks

Our secure cloud-based image scaling/cropping framework [29] allows a datacenter to scale or crop a hidden image such that the secret scaled/cropped image is recoverable from the hidden scaled/cropped images. The core idea behind this framework is to use a $(3, k, n)$ ramp secret image sharing based on Shamir's secret sharing [30], which is homomorphic to the addition and scalar multiplication operations [22] used in the integer version of the bilinear scaling operation, to share an image at the server side. To support image cropping without sending unwanted data to the user, we preserve the pixel positions of the secret image in the shadow image; and to support bilinear scaling, we do not use an additional non-homomorphic cryptosystem such as AES or watermarking in conjunction with ramp secret sharing. To remove the correlation among pixels in a shadow image, we use at least one random number in a secret sharing polynomial. The shared images (also called shadow images), are then transmitted to the datacenters. Upon request from a user, a datacenter scales/crops its shadow image, and sends the scaled/cropped image to the user. The user, upon receiving at least k scaled/cropped shadow images, recovers the secret scaled/cropped image.

Our secure pre-classification volume ray-casting framework [27] [31] hides the color of volume data from datacenters, and renders a color-hidden image from color-hidden data. The user, upon receiving at least k hidden rendered images, recovers the secret rendered image. As Shamir's secret sharing is non-homomorphic to multiplication operations, this scheme, however, cannot hide the opacities from the rendered images – therefore, disclosing the shape of the object. In this work, we address the incompatibility issue of volume ray-casting with Shamir's secret sharing by both modifying secret sharing and modifying volume ray-casting. We call the former approach *Secure Rendering by Modification of Shamir's Secret Sharing* (SR-MSSS), and the latter approach *Secure Rendering by Modification of Pre-classification Volume Ray-casting* (SR-MPVR). Both these techniques, by creating three different color shares for the red, green, and blue color components, and

by representing the share of a color component by a floating point number or by a large integer, however, incur high data overhead to the user. For applications requiring minimal overhead at the cost of high security, we propose a third technique called *Secure Rendering by Ramp Secret Sharing* (SR-RSS) that improves upon SR-MSSS by first replacing modified Shamir's secret sharing with a modified $(3, 4, 5)$ ramp secret sharing to create only one share for red, green, and blue colors, and then restricting the value of a share (which is a floating point number) to a smaller number and representing the restricted value with an integer.

Finally, assuming that Gouard Shading is used in ray-casting, we propose our secure cloud-based rendering framework [32] that not only uses the popular post-classification volume ray-casting to render volume data but also hides both the color and shape of an object from a datacenter. The core idea is to distribute the ray-casting tasks among two groups of datacenters such that even though rendering operations other than addition and scalar multiplication are not hidden, none of the groups can know the volume data and rendered image. To hide the parts of the volume data/image that are added and scalar multiplied, we use Shamir's secret sharing. In this framework, a server first performs the pre ray-projection operations of post-classification volume ray-casting, and then creates n shares of the scalar values and n shares of the outputs of the pre ray-projection operations (such as gradients and Phong illumination factors) using Shamir's (k, n) secret sharing. The shared information is then sent to the first group of datacenters called the *Interpolator*. According to a user's request, the Interpolator first interpolates the shared scalars, shared gradients, and shared Phong illumination factors; and then, by hiding the pixel positions, outsources the remaining volume ray-casting operations, such as classification, shading, and composition, to the second group of datacenters called the *Compositor*. After completing ray-casting, the Compositor transmits the hidden image to the client, who recovers the secret image by recovering the secret pixel coordinates and the secret colors of the pixels.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we review previous works related to ours and provide an overview of the techniques that we use in our work. Chapter 3 addresses

the incompatibility of Shamir's secret sharing with floating point operations. Chapter 4 discusses secure cloud-based image scaling/cropping framework. Chapter 5 discusses secure cloud-based pre-classification volume ray-casting framework, and Chapter 6 discusses secure cloud-based post-classification volume ray-casting framework. Chapter 7 concludes the work and proposes future direction for further research

Chapter 2

Background and Related Work

In this chapter, we first review the research works related to cloud-based imaging, cryptographic imaging, computation on hidden domain, and secure multi-party computation, and then provide an overview of image scaling and volume data rendering algorithms.

2.1 Cloud-based Imaging

Recently, cloud-based imaging, due to its promise of better services such as low cost, high scalability, availability, disaster recoverability etc., has drawn the attention of both academic researchers and enterprises. The main application of this technique has been in the field of medical imaging [10, 11, 12, 13, 33, 34, 35, 36, 36], where a datacenter is being used to store and process the data/images of a patient. In the following sections, we provide a brief overview of cloud-based medical imaging techniques.

2.1.1 Cloud-based data/image storage

To efficiently and cost effectively store medical images, researchers and enterprises are proposing to move the entire picture archiving and communication system (PACS) of a hospital to cloud datacenters [13, 33, 34, 35, 36]. For example, using Microsoft Windows Azure, Teng et al. proposed a cloud-based PACS system that uses a DICOM sever to handle store/query/retreive requests, a DICOM image indexer to parse the metadata and store them in a SQL Azure database, and a web

UI to allow searching and viewing of archived images [33]. Similarly, enterprises such as AT&T, Accenture, FreedomPACS, SCImage etc. are offering cloud-based PACS systems.

Parallelly, the security and privacy issues of cloud-based image storage [13, 14, 37, 38] have also been addressed in two possible scenarios: when a single datacenter is used, and when multiple datacenters are used. In the case of the use of a single datacenter, public key encryption techniques, such as watermarking, or chaos-based encryption, have been applied to protect the data/image [15, 17, 39, 40]; and in the case of the use of multiple datacenters, a secret sharing scheme has been used to distribute the secrecy among more than one datacenter [17]. For a complete list of existing cryptographic cloud storage systems, the reader can refer to AlZain et al.'s work [17], which concludes that the secret sharing based cloud storage systems are more secure than the encryption based systems.

2.1.2 Cloud-based data/image processing

Similar to cloud-based data/image storage, cloud-based data/image processing is also a growing trend. Researchers and enterprises are actively proposing cloud-based volume data rendering frameworks [10, 11, 12, 34] to render volume data. For example, using Azure cloud, Dorn et al. [11] proposed an adaptive data rendering framework that, according to requirements, performs volume ray-casting either in a cloud datacenter or at the client. By echoing the concerns of scalability in server-side rendering and resource availability in client-side rendering, Vazhenin proposed yet another cloud-based rendering framework [12]. Similarly, enterprises such as Sinha system [10], KDDI Inc. [41], etc. have started offering cloud-based volume data rendering frameworks to hospitals.

To the best of our knowledge, research on the security and privacy issues in cloud-based volume data/image processing, however, is a little explored area.

2.2 Cryptosystems Applied on Image

In this section, we review existing cryptographic imaging techniques to find their ability to meet our objective. In the next section, we survey existing volume data hiding techniques.

So far, watermarking techniques [42, 43, 44], chaos-based encryption [45, 46], and visual secret sharing schemes [47, 48] have been used to protect an image. Among these techniques, watermarking [44] and chaos-based encryption (which uses permutation for the chaos) cannot support arbitrary cropping of an image. Therefore, we exclude them from further study. Visual secret sharing, which secret shares an image among n participants either by visual cryptography [48] or by the application of threshold secret sharing schemes, however, can support cropping by hiding the color of each pixel independently.

In the following, we review existing visual cryptography and threshold secret sharing based image hiding schemes. Since three threshold secret sharing schemes, Shamir's secret sharing [30], Chinese remainder theorem-based secret sharing [49, 50], and Blakley secret sharing [23], are popular, we summarize each of them.

2.2.1 Visual cryptography

Visual cryptography, which was first proposed by Naor and Shamir [48], secret shares a binary image by using two boolean matrices, one for a white pixel and other for a black pixel. A row in a matrix acts as a share. The matrices are designed in such a way that the hamming weight of OR-ed k rows (i.e., bitwise OR of k rows) of the black-pixel-matrix is more than a threshold, and the hamming weight of OR-ed k rows of the white-pixel-matrix is less than the threshold. Since the threshold determines the transparency of a pixel in the secret image, one can know a secret color by knowing k or more rows of a matrix. The hamming weight of the vector obtained by OR-ing less than k rows of the black-pixel-matrix, however, is the same as the hamming weight of the vector obtained by OR-ing less than k rows of the white-pixel-matrix. Therefore, no information about a secret color can be found from less than k rows.

Visual cryptography is very easy to implement, and can even reconstruct an image without requiring a computer. Due to the use of binary matrices, visual cryptography, however, cannot support the scaling of shadow images, since the addition of color shares, as required by the scaling operation, can produce undefined interpolated values. For example, in $(2, 2)$ visual cryptography, if we distribute a white pixel over two rows: $\{(1, 1), (1, 1)\}$ and a black pixel over two rows: $\{(1, 0), (0, 1)\}$, then the addition of the color of the two pixels will produce undefined shares: $(10, 1)$ and

$(1, 10)$. Furthermore, the quality of the recovered image in visual cryptography is also a concern as it approximates the secret image. Therefore, even though visual cryptography has been extended to color images [51], we choose to exclude it from our work.

2.2.2 Blakley's secret sharing, and its application in sharing an image

Blakley's (k, n) secret sharing exploits a common geometric property that the intersection point of any $(k-1)$-degree non-parallel hyperplanes can be found only when k or more hyperplanes are known. Therefore, if the secret(s) is/are hidden as the coordinate(s) of the intersection point, then at least k hyperplanes, each of which can serve as a share, are required to know the secret. In the following, we provide a brief overview of the share distribution step and the secret reconstruction step.

Share distribution

Given k integers $(x_1, x_2, \ldots, x_k)$, one or more of which are the secret, the j^{th} hyperplane for all $1 \leq j \leq n$, is defined as $\sum_{i=1}^{k} x_i a_{ij} = b_j$, where each a_{ij} is a random number. Each j^{th} hyperplane then serves as the j^{th} share.

Secret reconstruction

Given the equations of any k hyperplanes, the coordinates of the intersection point of the hyperplanes (and hence the secret) are found by solving k equations: $\sum_{i=1}^{k} x_i a_{ij} = b_j$, where $1 \leq j \leq k$.

Using k colors as k coordinates of the intersecting points, researchers such as Tso [52] and Bozkurt et al. [53] have proposed secret image sharing schemes based on Blakley's scheme. Unlike other secret image sharing schemes, secret image sharing based on Blakley's scheme, however, is not popular as Blakley's scheme is not space efficient. In Blakley's scheme, a participant must store the a_{ij}'s and the b_j independently. Therefore, we will exclude this scheme from our future discussion.

2.2.3 Secret sharing methods based on the Chinese Reminder Theorem, and their application in sharing an image

To understand Chinese reminder theorem-based secret sharing, let us first understand how the Chinese reminder theorem works.

The Chinese reminder theorem states that for a set of pairwise relatively prime moduli $(q_1, q_2, \ldots, q_n)$, there exists a unique integer x for any given n residues $(r_1, r_2, \ldots, r_n)$ such that the congruences

$$\begin{aligned} x &= r_1 \bmod q_1 \\ x &= r_2 \bmod q_2 \\ &\vdots \\ x &= r_n \bmod q_n \end{aligned}$$

are satisfied. Thus, to find the value of x, first, the products of q_i's are calculated as $q = \prod_{i=1}^{n} q_i$; second, for each i, the multiplicative inverse of q_i, denoted as I_i, is calculated as $I_i = (\frac{q}{q_i})^{-1} \bmod q_i$; and finally, x is calculated as $x = \sum_{i=1}^{n} r_i \times \frac{q}{q_i} \times I_i$.

Using the Chinese reminder theorem, there are two main secret sharing schemes: Mignotte's scheme [49] and Asumuth Bloom's scheme [50]. Based on these two versions of secret sharing, there are two corresponding secret image sharing schemes.

Mignotte's secret sharing scheme

Mignotte's (k, n) secret sharing scheme uses a special sequence of moduli $q_1, q_2, \ldots q_n$ called Mignotte's sequence that satisfies the condition $q_1 < q_2 < \cdots < q_n$, and the condition $\prod_{i=0}^{k-2} q_{n-i} < \prod_{i=1}^{k} q_i$. The secret S, which satisfies the condition $\prod_{i=0}^{k-2} q_{n-i} < S < \prod_{i=1}^{k} q_i$, is then shared by the equation $S_i = S \bmod q_i$, where S_i is the i^{th} share for all $1 \leq i \leq n$. Given any k shares, the secret S is reconstructed using the Chinese reminder theorem on k equations: $x = S_i \bmod q_i$, where $1 \leq i \leq n$.

Using Mignotte's scheme, Jian and Chen [54] proposed a secret image sharing scheme that first XOR-ed the color of each pixel with a random number to break the spatial coherence, and then ap-

plied Mignotte's scheme on the XOR-ed color value. To reconstruct a secret, this scheme, therefore requires the random seed (to obtain the number that was XOR-ed with the secret) that was used during share creation phase. Furthermore, this scheme is not a perfectly secure scheme [55]. In other words, this scheme can disclose some information about the secret to a group of less than k participants. Therefore, we do not consider this scheme for our frameworks.

Asumuth Bloom's secret sharing scheme

Similar to Mignotte's, Asmuth and Bloom's (k, n) secret sharing also uses a special sequence of moduli $q_0, q_2, \ldots, q_n$, called the Asmuth Bloom sequence, that satisfies the condition $q_0 \prod_{i=0}^{k-2} q_{n-i} < \prod_{i=1}^{k} q_i$. Given a secret $S \in \mathbb{Z}_{q_0}$, the shares of S are then obtained by the equation $S_i = (S + \alpha.q_0) \bmod q_i$ (for all $1 \leq i \leq n$), where α is a random number satisfying $S + \alpha.q_0 \leq q_1 q_2 \ldots q_k$. Given any k shares S_1, S_2, …, S_k, the secret S is obtained by applying the Chinese reminder theorem on k sets of equations: $x = S_i \bmod q_i$, for each $1 \leq i \leq n$.

Based on Asmuth and Bloom's scheme, Ulutas et al. [56] proposed an image sharing scheme that broke the spatial coherence of an image by using different values of α to share the color values of different pixels. Although this scheme does not need random seeds and is perfectly secured, it is infrequently used as it is difficult to implement.

2.2.4 Shamir's secret sharing, and its application in sharing an image

Shamir's (k, n) (where, $k \leq n$) secret sharing is based on mathematical interpolation. To hide a secret, this scheme uses a $(k-1)$-degree polynomial whose zero-th degree coefficient is the secret. Using this polynomial, n shares of the secret are created by assigning n different values to the variable in the polynomial. Each share is then sent to a participant. When the shares of at least k participants are known, the polynomial is reconstructed by Lagrange interpolation, and the secret is found. In the following, we provide a mathematical overview of the share distribution and secret reconstruction steps. Next, we review some properties of Shamir's secret sharing.

Share distribution

Given a prime number q and a secret $a_0 \in \mathbb{Z}$ (where, $a_0 < q$), this step first creates the secret sharing polynomial

$$F(x) = \left(a_0 + \sum_{i=1}^{k-1} a_i x^i\right) \bmod q,$$

where $a_i < q$ is a random number in GF(q). Using this polynomial, any p^{th} (where, $0 < p < q$) share of a_0 is then found by

$$F(p) = \left(a_0 + \sum_{i=1}^{k-1} a_i p^i\right) \bmod q.$$

Secret reconstruction

Given k distinct share numbers $\{x_0, x_1, \ldots x_{k-1}\}$ and shares $\{y_0, y_1, \ldots y_{k-1}\}$ such that

$$y_i = F(x_i),$$

this step first finds the Lagrange interpolation polynomial $L(x)$ by

$$L(x) = \sum_{i=0}^{k-1} y_i t_i(x) \bmod q,$$

where

$$t_i(x) = \prod_{j=0, j\neq i}^{k-1} \frac{x - x_j}{x_i - x_j}$$

is called the Lagrange basis function. By the Unisolvence theorem, $L(x) = F(x)$. Thus, the secret a_0 can be obtained by setting $x = 0$ in $L(x)$.

Theorem 1. *(Unisolvence Theorem) Given k points* $\{(x_0, y_0),$ $(x_1, y_1), \ldots, (x_{k-1}, y_{k-1})\}$ *in GF(q) with mutually different* x_i*, there exists a unique polynomial* $L(x) \in GF(q)[x]$ *of at most* $k-1$ *degree such that* $L(x_i) = y_i$*,* $0 \leq i \leq k-1$.

Properties

Shamir's secret sharing is homomorphic to addition and scalar multiplication [22] – the two basic operations required in image processing. In other words, multiple secrets can be combined by direct addition and/or scalar multiplication on their shares. For example, if the participants are holding shares of a set of secrets $S = \{S_1, S_2, ..., S_j\}$, then without communicating amongst themselves, they can compute the shares of the secret $\sum_{i=1}^{j} I_i S_i$, where I_i is an integer.

Similar to Blakley's scheme and Asumuth Bloom's scheme, Shamir's scheme is perfectly secured (i.e., any combination of less than k shares disclose zero information about the secret). Mignotte's scheme is not perfectly secured. Similar to Asumuth Bloom's scheme, Shamir's scheme is an ideal secret sharing scheme, since the size of a share can be restricted to be equal to the size of the secret. Blakley's scheme is not an ideal secret sharing scheme. With comparison to Asumuth Bloom's scheme, Shamir's scheme requires less number of operations in secret reconstruction phase. Therefore, we use Shamir's secret sharing in our frameworks.

Shamir's secret sharing has also been used to securely multiply or divide a fixed number of shares [28, 57]. Typically, the division is performed as multiplication using the Newton Raphson method or Goldschmidt's scheme. Since our framework cannot fix the number of multiplications beforehand, we, however, do not use these schemes.

To protect a secret, Shamir's (k, n) secret sharing, requires disk space of n times the size of a share (as a share's size is equal to the secret's size).

To decrease the high storage requirement, a variant of Shamir's secret sharing called ramp secret sharing (or multi-secret sharing) is used [58, 59]. Ramp secret sharing uses l secrets as l coefficients in a secret sharing polynomial, and therefore decreases the size requirement by $\frac{1}{l}$ times. Thus, ramp secret sharing is typically used in secret image sharing. Ramp secret sharing, however, is not a perfectly secure scheme [58, 59], and it provides a tradeoff between the size and the security: the higher the value of l, the smaller the size of the resulting shares and the lower the level of security, and vice-versa.

Shamir's secret sharing, however, uses the modular prime operation, and therefore can neither share a floating point number nor perform floating point operations on the shares.

This issue can be addressed by either of two approaches: by omitting the modular prime operation from Shamir's secret sharing, or by representing a floating point number as a fixed point number (e.g., by first rounding off the floating point number by d decimal places, and then multiplying 10^d by it). The former approach is parallelly proposed by ourselves [27] and Finamore [26], and the latter is proposed by Catrina et al [28]. The use of any of these scheme, however, introduces a tradeoff as the exclusion of the modular prime operation from secret sharing weakens the security, and the fixed point representation of a floating point number introduces round-off error.

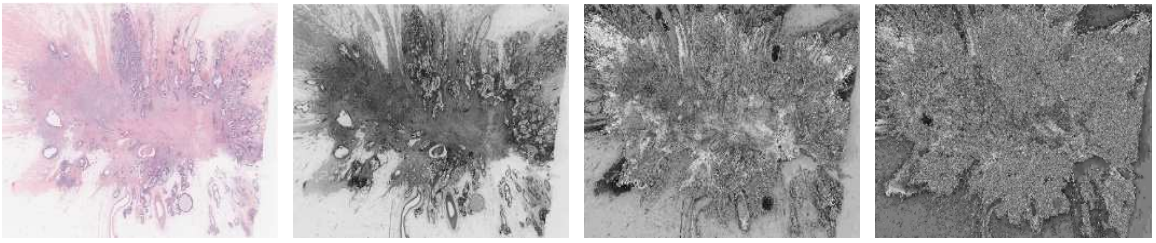
Alternatively, Chor et al. proposed a secret sharing scheme that can share a floating point number [60] without producing any side-effects. The main idea of this scheme is to share a secret $S \in \mathbb{R}$, where $S_{min} \leq S \leq S_{max}$, to n shares $S_1, S_2, \ldots, S_n$ in such a way that for each $1 \leq i \leq n-1$, each $S_i \in \mathbb{R}$ is randomly chosen from the interval $[S_{min}, S_{max}]$, and S_n satisfies $\sum_{i=1}^{n} S_i = S \pmod{S_{max}}$. This scheme, however, is non-homomorphic to floating point additions and scalar multiplications.

Secret image sharing based on Shamir's scheme

Secret image sharing based on Shamir's secret sharing is a thoroughly studied area [47, 61, 62, 63, 64, 65, 66, 67]. However, existing works assume that a participant (a shadow image holder) does not process the stored shadow image, and therefore focus on two main issues: how to decrease the size and how to increase the security of a shadow image. To decrease the size of a shadow image, (k, k, n) ramp secret sharing [58], which uses k color values $\{C_0, C_1, \ldots, C_{k-1}\}$ as secrets in a $k-1$ degree Shamir's secret sharing polynomial $F(x)$ as

$$F(x) = \sum_{i=0}^{k-1} C_i x^i \bmod q$$

(where q is a prime number), has been proposed [47, 61, 62, 63, 64]. The use of a (k, k, n) ramp secret sharing technique, however, can disclose the spatial coherence of the secret image in a shadow image (as shown in Figure 2.1), as a number of $F(x)$'s defined from a set of coherent color values can produce similar results [61]. Therefore, researchers have proposed to couple (k, k, n) ramp secret sharing with permutation [47], chaotic map [62], stenography [63] and matrix projection [61]



(a) (b) (c) (d)

Figure 2.1: Shadow images created by the $(3, 3, 5)$ ramp secret sharing technique that uses the R, G, B values of a pixel in $F(x)$: (a) is the secret image; and (b), (c), (d) are the 1^{st}, 2^{nd}, 3^{rd} shadow images, respectively. As $F(x)$ maps the R, G, B values to only one value, which is the color of a pixel of the shadow image, the shadow images are gray colored.

etc. to increase the security of the shadow image. The integration of these techniques with secret sharing, however, can destroy the homomorphic property of secret sharing (for example, when matrix projection or stenography are used) or randomize the pixel positions of the secret image (for example, when permutation or chaotic map are used). To hide the spatial coherence without using an additional technique, Alharthi and Atrey [64] recently proposed to use different share numbers to share the colors of different pixels. Similar to the other methods, the resulting secret sharing scheme is no longer homomorphic.

2.3 Cryptosystems Applied on Volume data

Research in hiding volume data is not as widespread as the hiding of images. So far, non-homomorphic digital watermarking techniques based on 3D-DWT and 3D-DCT [68], and the secret sharing technique [69] have been mainly proposed to hide volume data. However, even if we can use other cryptographic techniques such as AES, DES, ElGamal cryptosystem etc., unlike secret sharing, none of them are unconditionally secure, homomorphic to addition and scalar multiplication, and result in less computational overhead simultaneously [20].

2.4 Computation in Hidden Domain

Evolution in cloud computing has made computation in hidden domain a necessity highly desirable. Ideally, this requirement can be met when outsourced data/images are protected by a cryptosystem

that is homomorphic to the performed computations. For example, in theory, Gentry's lattice-based fully homomorphic scheme [70] can secure any cloud-based computation. However, this scheme is far from being used in practice as it's implementation is inefficient [18, 19, 71], and it cannot guarantee the correctness of computations [71]. As a result, researchers are proposing application-specific solutions.

Processing in hidden domain mainly can be divided into two groups: schemes using secure multi-party computation, and schemes those do not use secure multi-party computation.

The secure multi-party computation-based schemes distributes the processing among a number of participants in such a way that none of the participant can know more information than it requires. Using this method, Li et al. studied how to search encrypted cloud data [72, 73, 74], Wang et al. proposed a scheme to securely outsource linear programming to cloud datacenters [75], and various other researchers studied the feasibility of securing cloud-based e-voting, data mining, auctions etc [76, 77, 78, 79].

The non-secure multi-party-based schemes typically use somewhat homomorphic cryptosystems to secure operations that are homomorphic to the used cryptosystem. For example, Erkin et al. used two semantically secure additively homomorphic public-key encryption schemes: Paillier cryptosystem and DGK cryptosystem, to perform face recognition in hidden domain [80]. Their scheme involves two parties: Alice and Bob, where Alice owns a face image and Bob owns an image database. Both Alice and Bob want to run a face recognition algorithm to determine whether Alice's face image matches with any image in the Bob's database without disclosing information to one another. This scheme, however, is less suitable for third-party outsourcing since Bob, the third-party service provider, must interact with Alice, the server, during the execution of non-homomorphic operations. This shortcoming, however, can be avoided by exploiting the property of a specific algorithm, and by using simple data hiding techniques, such as obfuscation, in addition to a somewhat homomorphic cryptosystems. For example, Ayday et al. proposed a distributed architecture that uses Paillier cryptosystem and obfuscation to distribute the processing of genomic data among a number of participants such that none of the participant can get enough information to identify the owner of data [81]. Their solution, however, is similar to secure multi-party computation.

As our work is based on secure multi-party computation, we will review it in the next section.

Readers interested in other schemes, and willing to get a list of applications that are using hidden domain processing, are referred to a recent tutorial by Lagendijk et al. [82].

2.5 Secure Multi-Party Computation

In the early eighties, Yao first introduced the concept of secure multi-party computation by introducing the *Millionaire Problem* [83], which recognized the richest millionaire among two millionaires without disclosing the wealth of either of the millionaires. This work, although only limited to the comparison operation, introduced the concept of secure multiparty computation: the problem of distributing the computation of a function among n participants such that none of the participants can get more information than their input and the output of the computation. In a follow up work, Yao proposed a grabbled circuit-based generalized two-party secure computation protocol [84] that served as the basis for numerous further research [85, 86, 87, 88, 89, 90, 91, 92]. This research can be classified into two main categories [89]: secret sharing-based schemes and binary circuit-based schemes.

The secret sharing-based schemes distribute the secret among a number of participants, and allow simple arithmetic operations such as addition and scalar multiplication on the hidden data. These schemes are mainly used in secure e-voting [85], threshold signature [86], data mining [87] etc. This approach has also been successfully applied to securely auction farmers' bids in the Danish sugar beet industry [88].

On the other hand, the binary circuit-based approaches [84, 91], which are so far of interest to theorists, work on the principle of first representing any function as a binary circuit (a collection of logical gates), and then securely computing the binary circuit. To understand how the computation of a binary circuit is secured, let us take the example of Yao's grabbled construction for a single gate. Assume that one input of a gate is held by Participant A, and another is held by Participant B. Yao's construction hides the input of Participant B from Participant A by not disclosing Participant B's input to Participant A, and hides the input of Participant A from Participant B by encrypting Participant A's input. The output is also double encrypted using the input-keys (i.e., the cryptographic keys which replace the actual inputs) of both Participant A and Participant B as public keys. In this

construction, Participant A first finds cryptographic keys for all four possible inputs of the gate, and then uses these keys to double encrypt the real outputs. The encrypted outputs and the encrypted inputs are then permuted to hide the input order, and the permuted table is sent to Participant B with the input key of Participant A. Finally, Participant B decrypts the output using their input-key and Participant A's input-key.

Binary circuit-based schemes, however, are inefficient for arithmetic operations as they are performed in the binary domain. These schemes, can however, efficiently perform the comparison operations as required in secure online auctioning, verification of the correctness of outsourced computations, etc. Therefore, researchers have recently started to propose practical binary circuit-based secure multiparty computation schemes [90, 93, 89].

However, to the best of our knowledge, secure multi-party computation has not yet been applied for image scaling/cropping or scientific visualization. In this thesis, we are the first to study the feasibility of using Shamir's secret sharing-based secure multiparty computation for image scaling/cropping and for volume ray-casting, and are among a handful of researchers to propose practical secure cloud-based multiparty frameworks.

2.6 Volume Data Rendering and 2D Image Scaling

In this section, we will review image scaling and volume data rendering techniques since we use them in our work.

2.6.1 Image scaling

An image is typically scaled by bilinear or bicubic interpolation. As our work uses bilinear interpolation, we summarize it below.

Bilinear interpolation

Given the color values $\{C_0, C_1, C_2, C_3\}$ of any four pixels and two scaling factors $0 \leq h \leq 1$ and $0 \leq w \leq 1$ along the height and width of the image, bilinear interpolation finds the interpolated

color C by

$$C = \sum_{i=0}^{3} C_i D_i, \tag{2.1}$$

where $D_0 = (1-w)(1-h)$, $D_1 = (1-w)h$, $D_2 = w(1-h)$, and $D_3 = wh$. We call D_i the i^{th} interpolating factor.

2.6.2 Volume data rendering

Volume rendering, or 3D data rendering, renders 3D image either from a 3D volumetric data or from a set of 2D images. Depending on the input, existing 3D rendering algorithms are classified into two main categories: direct volume rendering, which inputs volume; and surface rendering, which inputs a 2D image set. This thesis focuses on direct volume rendering as this technique produces better quality images [94]. Among the direct volume rendering techniques, such as volume ray-casting, splatting, and shear wrapping, volume ray-casting is commonly used [8].

The main idea behind volume ray-casting is to project rays from each pixel of the image space on a 3D object, and find the color and opacity along each ray by mapping the physical properties of the object to optical properties (i.e., color and opacity). This idea is executed in a number of independent rendering components such as: gradient and normal estimation, classification, shading, ray-projection, sampling, interpolation, and composition. These components are typically arranged in two different pipelines: the pre-classification volume ray-casting pipeline and the post-classification volume ray-casting pipeline [95] [96].

Pre-classification volume ray-casting

Pre-classification volume ray-casting renders given volume data $V = \{v_{i,j,k} | v_{i,j,k}$ is the ijk^{th} data voxel$\}$ in the pipeline: gradient and normal estimation, classification, shading, ray-projection, sampling, interpolation, and composition.

Gradient and Normal Estimation: Given the scalar value $P_{v_{i,j,k}}$ of $v_{i,j,k}$, this step finds the gradient $G_{v_{i,j,k}} = (G^x_{v_{i,j,k}}, G^y_{v_{i,j,k}}, G^z_{v_{i,j,k}})$ and the normal $\mathcal{N}_{v_{i,j,k}}$ of $v_{i,j,k}$ by

$$G^x_{v_{i,j,k}} = \frac{P_{v_{i+1,j,k}} - P_{v_{i-1,j,k}}}{2}$$
$$G^y_{v_{i,j,k}} = \frac{P_{v_{i,j+1,k}} - P_{v_{i,j-1,k}}}{2}$$
$$G^z_{v_{i,j,k}} = \frac{P_{v_{i,j,k+1}} - P_{v_{i,j,k-1}}}{2}$$

and

$$\mathcal{N}_{v_{i,j,k}} = \frac{G_{v_{i,j,k}}}{|G_{v_{i,j,k}}|}$$

respectively.

Classification: This step finds the color $C_v^{\uparrow}$ and the opacity A_v of a data voxel $v \in V$ from a given look-up table indexed by P_v.

Shading: Given the ambient coefficient k_a, diffuse coefficient k_d, specular coefficient k_s, specular shininess n, light direction L, and reflected light direction R, this step finds the shaded color of v as

$$C_v = C_v^{\uparrow} Y_v + Z_v$$

where

$$Y_v = k_a + k_d MAX(\mathcal{N}_v L, 0)$$

and

$$Z_v = k_s MAX\big((\mathcal{N}_v R)^n, 0\big)$$

are called Phong illumination factors.

Ray Projection: In this step, rays from the pixels of the image space are projected on V.

Sampling: This step samples a projected ray at c sample points $s_1, s_2, ..., s_c$.

Interpolation: This step computes the color C_s and the opacity A_s of a sample point s by interpolating the colors and the opacities of $N(s)$, eight neighbouring voxels of s. Mathematically, C_s and A_s are defined as

$$C_s = \sum_{v \in N(s)} C_v D_v, \tag{2.2}$$

and

$$A_s = \sum_{v \in N(s)} A_v D_v, \tag{2.3}$$

where $C_v \in \mathbb{N}$ and $A_v \in \mathbb{R}$ are the color and opacity of v, and $D_v \in \mathbb{R}$ is the interpolating factor of v. The interpolating factor D_v is calculated from the xyz-coordinate of s and the xyz-coordinates of the voxels $v \in N(s)$.

We know that C_v satisfies

$$0 \leq C_v \leq 255, \tag{2.4}$$

and D_v satisfies

$$0 \leq D_v \leq 1, \tag{2.5a}$$

$$\sum_{v \in N(s)} D_v = 1. \tag{2.5b}$$

Thus, by putting Inequality 2.4 and Inequality 2.5 in Equation 2.2, we conclude

$$0 \leq C_s \leq 255. \tag{2.6}$$

Similarly, A_v is defined as

$$0 \leq A_v \leq 1. \tag{2.7}$$

Therefore, by putting Inequality 2.5 and Inequality 2.7 in Equation 2.3, we conclude

$$0 \leq A_s \leq 1. \tag{2.8}$$

Composition: In this step, the colors and opacities of the sample points along a ray are accumulated to produce the composited color and the composited opacity. Mathematically, the composited color C and the composited opacity A of the sample points $s_1, s_2, ..., s_c$ are defined as

$$C = \sum_{i=1}^{c} C_{s_i} O_i \tag{2.9}$$

and

$$A = \sum_{i=1}^{c} O_i, \tag{2.10}$$

where O_i is defined as

$$O_i = A_{s_i} \prod_{j=i+1}^{c} \big(1 - A_{s_j}\big). \tag{2.11}$$

By putting Inequality 2.8 in Equation 2.11, $O_i \in \mathbb{R}$ satisfies

$$0 \leq O_i \leq 1. \tag{2.12}$$

Furthermore, we know that $A \leq 1$. Therefore, by Equation 2.10, O_i also satisfies

$$\sum_{i=1}^{c} O_i \leq 1. \tag{2.13}$$

Now, by putting Inequality 2.6, Inequality 2.12, and Inequality 2.13 in Equation 2.9, we conclude

$$0 \leq C \leq 255. \tag{2.14}$$

Furthermore, by Inequality 2.9 and Inequality 2.12, composite color C satisfies $C \in \mathbb{R}$. We, however, know that the color of a pixel is a whole number. Therefore, the composite C is truncated to obtain the final color.

Post-classification volume ray-casting

Post-classification volume ray-casting renders a volume V in a pipeline: gradient and normal estimation, ray-projection, sampling, interpolation, classification, shading, and composition. When

Gouard shading is used, the Phong illumination factors, however, are calculated before the projection of the rays. We use Gouard shading. Thus, we will discuss post-classification volume ray-casting with Gouard shading.

Gradient and Normal Estimation: Similar to pre-classification, this step calculates the gradient G_v and the normal $\mathcal{N}_v$ of a voxel $v \in V$.

Calculation of Illumination Factors: Similar to pre-classification, this step calculates the Phong illumination factors Y_v and Z_v of v.

Ray Projection: In this step, rays from each pixel of the image spaces are projected on V.

Sampling: In this step, a projected ray is sampled at c sample points $s_1, s_2, ..., s_c$.

Interpolation: In this step, the scalar value and gradient of a sample point s are calculated by interpolating the scalar values and gradients of the eight neighbouring voxels of s. Mathematically, the interpolated scalar value P_s (the same as for the interpolated gradient G_s) is calculated as

$$P_s = \sum_{v \in N(s)} P_v D_v, \tag{2.15}$$

where $P_v \in \mathbb{N}$ is the scalar value of v and $D_v \in \mathbb{R}$ is the interpolating factor of v.

Classification: In this step, the classified color $C_s^{\uparrow}$ and the classified opacity A_s of s are found from the given look-up tables by using P_s and G_s as indices.

Shading: In this step, the shaded color C_s of s is found from the classified color $C_s^{\uparrow}$ by

$$C_s = C_s^{\uparrow} Y_s + Z_s \tag{2.16}$$

where

$$Y_s = \sum_{v \in \mathcal{N}(s)} Y_v D_v$$

and

$$Z_s = \sum_{v \in \mathcal{N}(s)} Z_v D_v.$$

Composition: Similar to the composition step of pre-classification volume ray-casting, this step computes the color C and opacity A along a ray by accumulating the colors and opacities of all c sample points.

2.7 Chapter Summary

In this chapter, we first discussed about the growing demand for practical secure cloud-based data/image processing systems, and the inability of exiting homomorphic cryptosystems to meet the requirement. Then, we argued that secure multi-party computation can be an alternative to homomorphic schemes to securely process data/image at a datacenter. To find the best suited secure multi-party computation scheme for our requirement, we reviewed the grabbled circuit method, Bakley's secret sharing method, Chinese Reminder Theorem-based secret sharing method, and Shamir's secret sharing method in detail, and chosen Shamir's secret sharing-based secure multi-part computation method. Finally, we provided a brief overview of three commonly used data/image processing algorithms: bilinear image scaling, pre-classification volume ray-casting, and post-classification volume ray-casting, which are considered in the thesis.

Chapter 3

Using Floating Point Numbers in Shamir's Secret Sharing

Shamir's Secret Sharing operates in a finite field by performing modular prime operations. Typically, an image processing algorithm, such as image scaling/cropping and volume ray-casting, performs floating point numbers. Since floating point numbers are incompatible with modular prime operations, Shamir's secret sharing cannot be used in conjunction with an image processing algorithm.

To address the incompatibility of Shamir's secret sharing with a floating point number, in this chapter, we discuss two approaches: the exclusion of the modular prime operation from secret sharing, and the conversion of a floating point number to a fixed point number. The former approach is parallelly proposed by us [27] and Finamore [26], and the later approach is proposed by Catrina et al. [28]. However, due to the exclusion of modular prime operation from secret sharing, there can be loss in security; and due to rounding a floating point number to convert it to an integer, there can be rounding error. We study the effect of removing modular prime operation from Shamir's secret sharing, and analyzes the rounding error in the later approach.

3.1 Exclusion of the Modular Prime Operation

Similar to the work of Finamore [26], we exclude the modular prime operation from Shamir's secret sharing to make it compatible with floating point operations. The modified secret sharing polynomial is defined as

$$F'(x) = a_0 + \sum_{i=0}^{k-1} a_i x^i,$$

and the modified Lagrange interpolation formula is defined as

$$L'(x) = \sum_{i=0}^{k-1} F'(x_i) t_i(x).$$

We use this modified secret sharing for our SR-MSSS and SR-RSS schemes of secure pre-classification framework (Chapter 5).

Due to the exclusion of the modular prime operation, a cryptosystem, however, no longer works in a finite field. Therefore, the modified cryptosystem can lose security. In the following, we discuss the security loss.

3.1.1 Security analysis of the modified Shamir's secret sharing

To analyze the security of the modified secret sharing method, one must first understand how Shamir's (k, n) secret sharing hides a secret.

To hide a secret, Shamir's secret sharing exploits the condition that to know a $(k-1)$-degree polynomial $F'(x)$, the knowledge of at least k $F'(x_i)$'s is required. The uncertainty in finding the polynomial from less than k $F'(x_i)$'s varies with the number of $F'(x_i)$'s. The higher the number of known $F'(x_i)$'s, the less uncertainty there is. For example, suppose that we are given a share $F'(x_i) = 9$ for a share number $x_i = 1$, and are told that $F'(x)$ is a second degree polynomial. Then, we can find the equation $\sum_{i=0}^{2} a_i 1^i = 9$, which is satisfied by 55 sets of (a_0, a_1, a_2). Thus, a_0 can take 55 possible values with the knowledge of $x_i = 1$ and $F'(1) = 9$. Now, suppose that we know another share $F'(x_i) = 22$ for the share number $x_i = 2$. From this knowledge, we get another equation $\sum_{i=0}^{2} a_i 2^i = 22$. In anticipation of knowing the coefficients a_i's, we now subtract two known equations, and get a new equation $a_1 + 3a_2 = 13$. This resultant equation is satisfied by

five possible pairs of (a_1, a_2), and for these values of (a_1, a_2), there exists a maximum of five a_0's satisfying any $F'(x)$. Thus, the secret can now take 5 possible values. To counter this fluctuation in range of possible values, Shamir's secret sharing uses the modular prime operation. By not using modular prime operation, we remove this security shield.

However, even without the modular prime operation, there exists uncertainty to knowing a secret in the modified secret sharing scheme. To obtain a relationship between this uncertainty with the knowledge of the value of k, the value of share number x_i, and the value of share $F'(x_i)$, we provide the following formulas.

Let us start with $k = 2$, i.e., from a first degree polynomial $F'(x) = a_0 + a_1 x$, and assume that one $F'(x_i)$ is known. Clearly, for each value of a_0, there exists only one $a_1 x_i$ that can satisfy the equation $F'(x_i) = a_0 + a_1 x_i$. Therefore, if we can obtain possible values of a_1 (note that x_i is already known), then we can obtain the number of possible values of a_0. We, however, know that both a_0 and a_1 are positive integers satisfying $0 \leq a_0 \leq F'(x_i)$ and $0 \leq a_1 x_i \leq F'(x_i)$. Thus, a_1 can only take values from the range [0, $\frac{F'(x_i)}{x_i}$]. As a result, the secret a_0 can take $\lfloor \frac{F'(x_i)}{x_i} \rfloor + 1$ possible values.

Similarly, we can obtain the possible values of the secret from $k = 3$ and the knowledge of one $F'(x_i)$ as

$$T = \sum_{a=0}^{\lfloor \frac{F'(x_i)}{x_i^2} \rfloor} \Big(\lfloor \frac{F'(x_i) - a x_i^2}{x_i} \rfloor + 1 \Big). \tag{3.1}$$

With an increase in the value of $k > 3$, the value of T increases as the combination of $k-1$ coefficients that satisfy a $k-2$ degree polynomial $F'^{,k-2}(x, S)$ is also part of the combination of k coefficients that satisfy the $k-1$ degree polynomial $F'^{,k-2}(x, S) + a_{k-1} x^{k-1}$.

Hence, the number of choices to know the secret can be sufficiently large. For example, even in the extreme case of $k = 2$, $x_i = 1$, and $F'(x_i) = 1$, the value of T is two: equal to the number of choices in the case of Shamir's original secret sharing.

Now let us consider the case when l number (where $1 < l < k$) of $F'(x_i)$'s are known. In this case, one can get a polynomial of $k - l$ degree by solving the $F'(x_i)$'s. Based on the above arguments, the resultant polynomial involves some degree of uncertainty. The degree of uncertainty, however, increases with an increase in the value of l.

The uncertainty of knowing $F'(x)$ from the knowledge of $F'(x_i)$ and x_i is higher when the value of $F'(x_i)$ is higher for a fixed x_i or when the value of x_i is lower for a fixed $F'(x_i)$. Therefore, we recommend using a smaller x_i and a higher a_i.

However, by not using a modular prime operation, the modified secret sharing may be prone to side-channel attacks. This thesis leaves the investigation of a side channel attack on the modified secret sharing as an open problem.

3.2 Modifying a Floating Point Number to a Fixed Point Number

In this approach, we propose to address the incompatibility issue of Shamir's secret sharing with a floating point number by converting the floating point numbers to fixed point numbers. A float a_0 can be converted to a fixed point number by first rounding a_0 by d decimal places, and then multiplying b^n (where, $b \in \mathbb{N}$) to the rounded off value. In this thesis, we choose $b = 10$. Catrina et al. [28] proposed a similar scheme by choosing $b = 2$.

In other words, to convert a floating point number a_0 to a fixed point number $a_0^{(d)}$, we first round-off a_0 by using Definition 1, and then multiply 10^d by $Round(a_0, d)$.

Definition 1. *If $R = I.N_1N_2N_3 \ldots N_d \ldots$, is a floating point number, where $0 \leq I \leq 9$, $0 \leq N_i \leq 9$, and $d \in \mathbb{N}$, then the value of $Round(R, d)$ is defined as*

$$Round(R,d) = \begin{cases} I.N_1N_2N_3 \ldots N_d, & \text{if } N_{d+1} < 5 \\ I.N_1N_2N_3 \ldots (N_d+1), & \text{if } N_{d+1} > 5 \\ I.N_1N_2N_3 \ldots N_d, & \text{if } N_{d+1} = 5, N_d \text{ is an even number, } \textbf{and} \text{ for each } t > d+1, N_t = 0 \\ I.N_1N_2N_3 \ldots (N_d+1), & \text{if } N_{d+1} = 5, N_d \text{ is an odd number, } \textbf{and} \text{ for each } t > d+1, N_t = 0 \\ I.N_1N_2N_3 \ldots (N_d+1), & \text{if } N_{d+1} = 5 \textbf{ and} \text{ there exists at least one } t > d+1 \text{ such that } N_t > 0 \end{cases}.$$

Thus, $a_0^{(d)}$ is defined as

$$a_0^{(d)} = (a_0 + \epsilon_d) \times 10^d,$$

where ϵ_d is the round-off error. This error is analyzed in the next section.

We use this scheme for our secure image scaling/cropping framework (Chapter 4), SR-MPVR scheme of secure pre-classification framework (Chapter 5), and secure post-classification framework (Chapter 6).

3.2.1 Error analysis

Proposition 1. *If ϵ_d is the error resulting from rounding off a given floating point number by d decimal places, then the lower bound of ϵ_d, which is -0.5×10^{-d}, is obtained when: (i) the value of the d^{th} decimal place digit of the floating point number is an even number, (ii) the value of the $(d+1)^{th}$ decimal place digit of the floating point number is five, and (iii) the value of each $(d+t)^{th}$ (where, $t > 1$) decimal place digit of the floating point number is zero.*

Proposition 2. *If ϵ_d is the error resulting from rounding off a given floating point number by d decimal places, then the upper bound of ϵ_d, which is 0.5×10^{-d}, is obtained when: (i) the value of the d^{th} decimal place digit of the floating point number is an odd number, (ii) the value of the $(d+1)^{th}$ decimal place digit of the floating point number is five, and (iii) the value of each $(d+t)^{th}$ (where, $t > 1$) decimal place digit of the floating point number is zero.*

Claim 1. *If K is scalar and R is a floating point number, then the error due to $K \times Round(R, d)$ is bounded by $\pm 0.5K \times 10^{-d}$.*

Claim 2. *If R_1 and R_2 are two floating point numbers, then the error due to $Round(R_1, d) + Round(R_2, f)$ is bounded by $\pm(0.5 \times 10^{-d} + 0.5 \times 10^{-f})$.*

Corollary 1. *If K_i and R_i are the i^{th} scalar and i^{th} floating point number respectively, then the error due to $\sum_{i=1}^{c} K_i \times Round(R_i, d)$ is bounded by $\pm 0.5 \times \left(\sum_{i=1}^{c} K_i\right) \times 10^{-d}$*

Proof. Proven by Claim 1 and Claim 2. □

3.3 Chapter Summary

In this chapter, we discussed two methods to address the incompatibility of the floating point operation of a data/image processing algorithm with the modular prime operation of Shamir's secret sharing. The first method removes the modular prime operation from secret sharing, and the second method converts a floating point number to a fixed point number by first rounding off the floating point number by d decimal places and then multiplying 10^d to the rounded off value. We showed that the exclusion of modular prime operation from secret sharing can weaken the security of secret sharing, and the rounding off a float to a fixed point number can involve rounding error.

Chapter 4

Secure Cloud-based Image Scaling/Cropping

In this chapter, we propose the secure cloud-based image scaling/cropping framework that hides important images at datacenters, but allows image scaling and cropping on noise-like hidden images such that the processed secret image can be recovered from the processed hidden images. This requirement has arisen from the trend of servers such as hospitals wanting to outsource the storage and the processing of images to third-party datacenters without disclosing the content of images.

A naive solution to our requirement would be for the server to create multiple secret images at different resolutions (to support scaling), and to divide each secret image into independently decodable tiles (to support cropping). Each tile could then be hidden by applying a cryptosystem, and the hidden tiles could be sent to datacenters. When users request a region of an image at a particular scale, each datacenter could send the hidden tiles that overlap with the region at the nearest resolution to the user. Such a solution, however, could cause additional data to be sent to the user.

Therefore, we decided to directly scale and crop a hidden image. We assumed that scaling is performed by bilinear interpolation. Thus, we required a cryptosystem that is homomorphic to addition and scalar multiplication. Hence, we used Shamir's (k, n) secret sharing. The main idea of our scheme is to secret share an image at the server side and send n noise-like shadow images to

n datacenters, allow the datacenters to perform image scaling and cropping operations on shadow images, and recover the processed secret image from k processed shadow images. As discussed in Section 2.2.4, existing secret image sharing schemes, however, cannot fulfill our requirement as they destroy the homomorphic property of Shamir's secret sharing. Therefore, we propose a new secret image sharing scheme.

4.1 A New Secret Image Sharing Scheme

To allow scaling on shadow images, we need to keep the homomorphic property of secret sharing, and to allow cropping without sending extra bits of data, we need to keep the pixel position of secret image intact in the shadow image. Therefore, unlike existing works, we cannot use an additional cryptosystem in conjunction with secret sharing. As a result, as shown in Figure 2.1, information about the image can be leaked. Such information loss occurs due to the fact that existing secret image sharing schemes cannot break the spatial coherence of an image.

We observed that the spatial coherence in a shadow image can be broken if correlation among the adjacent pixels of the secret image can be destroyed. The inter-pixel correlation can be destroyed if the results of the secret sharing polynomials of neighboring pixels can be different. Existing secret image sharing schemes do not fulfill this requirement as they use the colors of pixels as all the coefficients in a secret sharing polynomial. We propose to use at least one random number as a coefficient in our secret sharing polynomial.

Furthermore, we also observe that to facilitate dynamic cropping, the pixel positions of the secret image must be kept unchanged in a shadow image. Therefore, we do not use the color values of one pixel in the secret sharing polynomial of another pixel. As a result, we are left with three choices (as the color of a pixel is represented by three components: red (R), green (G), and blue (B)): the use of one color component, the use of two color components, or the use of three color components in the definition of a secret sharing polynomial. These options, which are different cases of ramp secret sharing, provide tradeoffs between the security and the size of the shadow image: the first being most secure (as it uses only one secret in the secret sharing polynomial [58]) but resulting in the largest shadow image size (as it creates three different shares per pixel by using three secret sharing

polynomials) and the last being the least secure but resulting the smallest size. We choose the last option and design the randomized ramp secret image sharing method by defining a polynomial $F(x)$ of a $(3, k, n)$ (where, $4 \leq k \leq n$) ramp secret sharing technique as

$$F(x) = (R + Gx + Bx^2 + \sum_{i=3}^{k-1} a_i x^i) \bmod q,$$

where, each a_i is a random number satisfying $1 \leq a_i < q$ for at least one i and $0 \leq a_i < q$ for other i's. As shown in Figure 4.3, randomized ramp secret image sharing produces noise-like shadow images.

Note that although randomized ramp secret image sharing is designed for color images, it can be adjusted for gray images by using the gray color as the only secret in $F(x)$.

4.1.1 Supporting bilinear scaling

Recall that although bilinear interpolation performs addition and scalar multiplication operations that are supported by Shamir's secret sharing homomorphism [22], it involves floating point operations, which are incompatible with the modular prime operation of Shamir's secret sharing. To make bilinear interpolation compatible with the modular prime operation, we therefore convert its floating point operations to fixed point operations by converting its floating point operands to fixed point operands. As discussed in Section 2.6.1, an interpolating factor D_i is the only floating point number in the bilinear interpolation. Therefore, we represent D_i by a fixed point number

$$D_i^{(d)} = (D_i + \epsilon_{D_i,d}) \times 10^d, \tag{4.1}$$

where $\epsilon_{D_i,d}$ is the round-off error.

By replacing each D_i with $D_i^{(d)}$ in Equation 2.1, we find the scaled color C'_s of a pixel s as

$$\begin{aligned} C'_s &= \sum_{i=0}^{3} C_i D_i^{(d)} \\ &= \sum_{i=0}^{3} C_i (D_i + \epsilon_{D_i,d}) \times 10^d \qquad \text{(by Equation 4.1)} \end{aligned}$$

$$= \Big(\sum_{i=0}^{3} C_i D_i + \sum_{i=0}^{3} C_i \epsilon_{D_i,d} \Big) \times 10^d$$
$$= (C_s + \epsilon_{C_s}) \times 10^d, \qquad \text{(by Equation 2.1)} \tag{4.2}$$

where C_s is the scaled color by conventional bilinear scaling, and

$$\epsilon_{C_s} = \sum_{i=0}^{3} C_i \epsilon_{D_i,d}$$

is the scaling error.

Given that the color of each pixel C_i satisfies $0 \leq C_i \leq 255$, by using Corollary 1, ϵ_{C_s} can be calculated to be $-510 \times 10^{-d} \leq \epsilon_{C_s} \leq 510 \times 10^{-d}$. Furthermore, we know that the scaled color C_s also satisfies $0 \leq C \leq 255$. Therefore we can conclude that C'_s satisfies

$$(255 - 510 \times 10^{-d}) \times 10^d \leq C'_s \leq (255 + 510 \times 10^{-d}) \times 10^d. \tag{4.3}$$

Note that by rounding off the interpolating factors by d decimal places, we restrict the maximum number of scaled images to $10^d - 1$. However, as the maximum number of decimal places that an interpolating factor can take is restricted in a computer, one can choose d to be large enough to cover all possible scaled images. Therefore, we can claim that the proposed modified bilinear interpolation provides a sufficiently large number of scaled images.

4.2 Scaling/Cropping an Image in Hidden Domain

We now describe how we will use the new secret image sharing scheme in scaling/cropping an image in hidden domain. Figure 4.1 shows the three components of our framework: the image source (which is the server), n datacenters, and the user. This framework has been designed with the assumptions that: (i) the image source and the user are trusted entities, (ii) the datacenters do not communicate confidential information among themselves, and (iii) an adversary cannot access more that $k - 1$ datacenters.

As shown in Figure 4.2, a typical workflow of storing and recovering an image can be divided

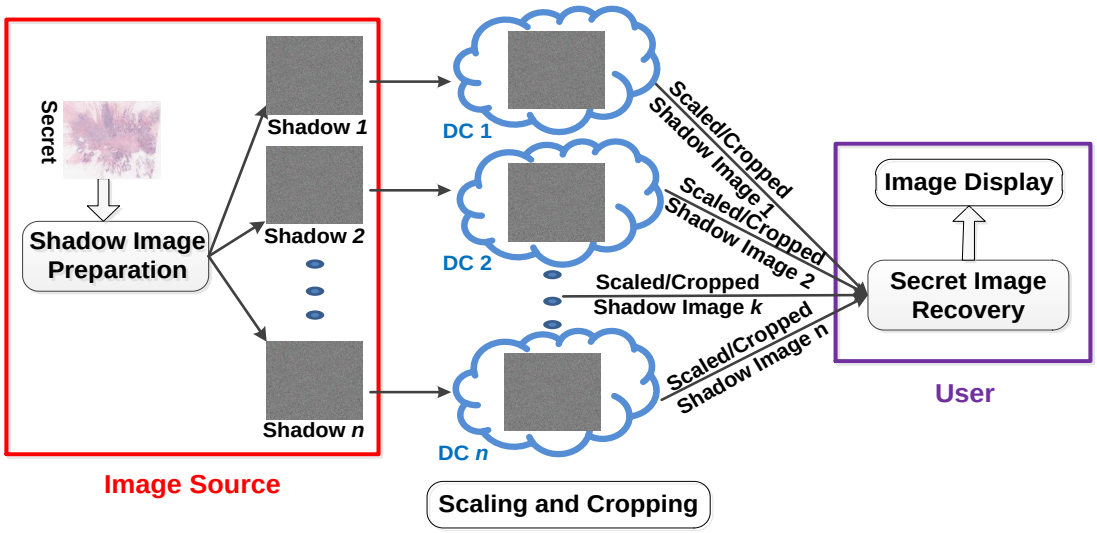


Figure 4.1: Secure cloud-based image scaling/cropping.

into four steps: (i) shadow image preparation, (ii) image scale/crop request, (iii) shadow image scaling/cropping, and (iv) secret image recovery.

4.2.1 Shadow image preparation

In the first step, the server creates n shadow images by applying our secret image sharing technique on the secret image. Thus, it computes $F_i(x)$ for each pixel i of the secret image for n different values of x.

To define a $F_i(x)$, we need to pick a prime number q that is greater than the reconstructed secret. In our case, the reconstructed secret, which can be C'_s (when the shared color values are interpolated), has the maximum value of $(255+510\times10^{-d})\times10^{d}$ (by Equation 4.3). Therefore, the server must choose q greater than $(255 + 510 \times 10^{-d}) \times 10^{d}$. Furthermore, the shares of the color values that are added together must belong to one GF(q). As the server has no prior knowledge of which of the shared colors will be added, it can use one q for all the polynomials. In finding the value of q, the server also needs to fix the value of d in this step.

Using q, the server defines a secret sharing polynomial $F_i(x)$ for its i^{th} pixel as

$$F_i(x) = \left(R_i + G_i x + B_i x^2 + \alpha_{i,x}\right) \bmod q, \tag{4.4}$$

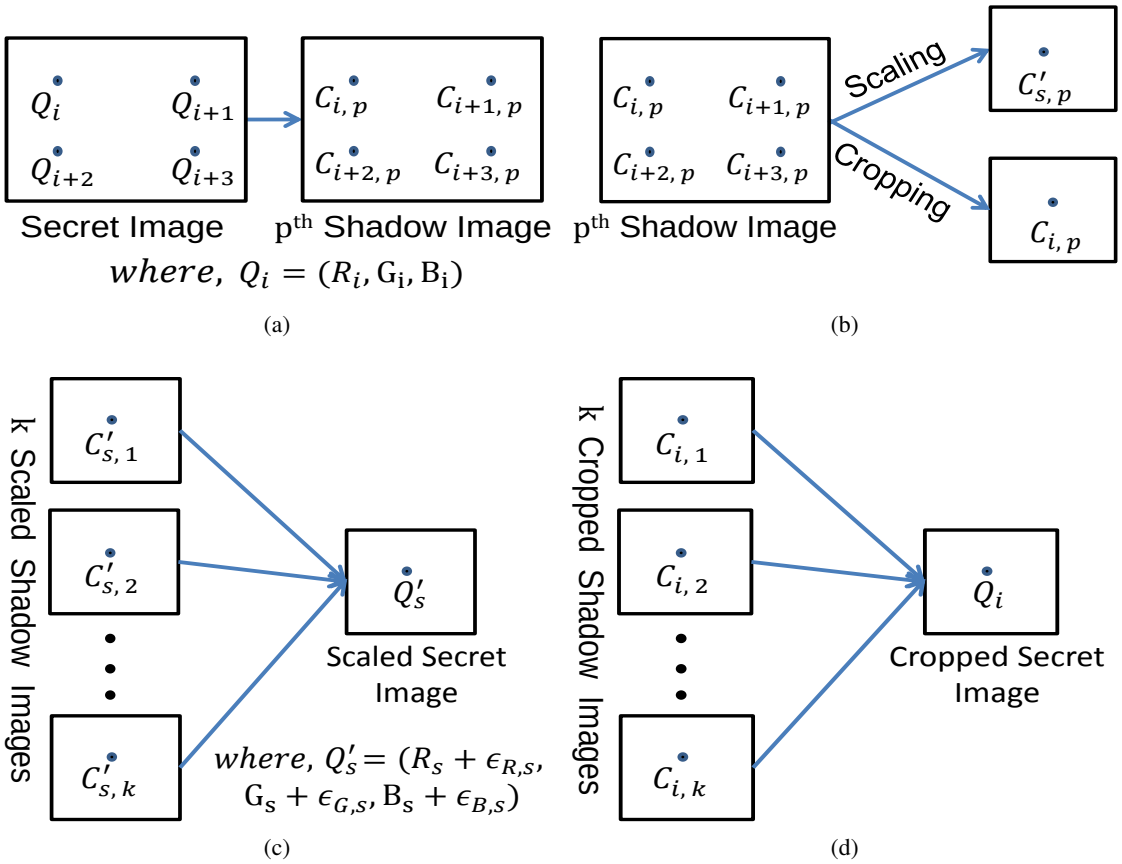


Figure 4.2: Workflow of secure cloud-based image scaling/cropping framework: (a) shadow image preparation, (b) scaling/cropping of p^{th} shadow image, (c) recovery of secret scaled image, (d) recovery of secret cropped image.

where R_i, G_i, and B_i are the colors of the pixel, $\alpha_{i,x} = \sum_{j=3}^{k-1} a_{i,j}x^j$, and $a_{i,j}$ are randomly chosen integers. Using this polynomial, the p^{th} share of all three colors can be found by

$$C_{i,p} = F_i(p) = \big(R_i + G_i p + B_i p^2 + \alpha_{i,p}\big) \bmod q, \tag{4.5}$$

where $\alpha_{i,p} = \sum_{j=3}^{k-1} a_{i,j}p^j$. The p^{th} (for all $1 \leq p \leq n$) shadow image is then constructed with $C_{i,p}$ as the color value of its i^{th} pixel. This shadow image is sent to the p^{th} datacenter. As shown in Figure 4.3, a shadow image is a noise-like image, and therefore, it does not disclose any information about the secret image to the datacenter.

4.2.2 Shadow image scaling/cropping

When a user requests to view an image, it will supply the datacenters with three parameters, scaling factors w and h (note that in practice $w = h$), and a rectangular region of interest I. Upon request, each datacenter scales a shadow image using modified bilinear interpolation and crops the shadow image by selecting the color values of only the pixels that are part of the requested region I (as shown in Figure 4.2b). As cropping is equivalent to the existing cropping technique, we will not discuss it further. For scaling, we will only discuss the scaling of the p^{th} shadow image that is performed by the p^{th} datacenter as the scaling of one shadow image is equivalent to the scaling of another.

To scale its shadow image, the p^{th} datacenter first computes $D_j^{(d)}$, the integer representative of the j^{th} interpolating factor D_j, from the scaling factors h and w, and the round-off parameter d. Then, depending on the value of h and w, the datacenter iteratively selects the color values $\{C_{i,p}, C_{i+1,p}, C_{i+2,p}, C_{i+3,p}\}$ of four pixels and interpolates them. Therefore, by Equation 4.2, the interpolated color $C'_{s,p}$ of the s^{th} pixel of the scaled image can be derived as

$$\begin{aligned} C'_{s,p} &= \sum_{j=0}^{3} C_{i+j,p} D_j^{(d)} \\ &= \sum_{j=0}^{3} \big((R_{i+j} + G_{i+j}p + B_{i+j}p^2 + \alpha_{i+j,p}) \bmod q\big) D_j^{(d)} \qquad \text{(by Equation 4.5)} \end{aligned}$$

$$\equiv \Big(\sum_{j=0}^{3} (R_{i+j} + G_{i+j}p + B_{i+j}p^2 + \alpha_{i+j,p}) D_j^{(d)} \Big) \bmod q$$

$$\equiv \Big(\sum_{j=0}^{3} R_{i+j} D_j^{(d)} + \sum_{j=0}^{3} G_{i+j}p D_j^{(d)} + \sum_{j=0}^{3} B_{i+j}p^2 D_j^{(d)} + \sum_{j=0}^{3} \alpha_{i+j,p} D_j^{(d)} \Big) \bmod q$$

$$\equiv \Big(\sum_{j=0}^{3} R_{i+j} D_j^{(d)} + \sum_{j=0}^{3} G_{i+j}p D_j^{(d)} + \sum_{j=0}^{3} B_{i+j}p^2 D_j^{(d)} + \sum_{j=0}^{3} \sum_{l=3}^{k-1} a_{i+j,l} p^l D_j^{(d)} \Big) \bmod q$$

$$\Big(\text{Substituting } \alpha_{i,p} = \sum_{j=3}^{k-1} a_{i,j} p^j\Big)$$

$$\equiv \Big(\big(\sum_{j=0}^{3} R_{i+j}(D_j + \epsilon_{D_j,d}) + \sum_{j=0}^{3} G_{i+j}p(D_j + \epsilon_{D_j,d}) + \sum_{j=0}^{3} B_{i+j}p^2(D_j + \epsilon_{D_j,d}) + \sum_{j=0}^{3} \sum_{r=3}^{k-1} a_{i+j,r} p^r (D_j + \epsilon_{D_j,d}) \big) \times 10^d \Big) \bmod q \tag{4.6}$$

$$\Big(\text{Substituting } D_j^{(d)} = (D_j + \epsilon_{D_j,d}) \times 10^d\Big)$$

$$\equiv \Big(\big(\sum_{j=0}^{3} (R_{i+j} D_j + R_{i+j} \epsilon_{D_j,d}) + \sum_{j=0}^{3} (G_{i+j} D_j + G_{i+j} \epsilon_{D_j,d}) p + \sum_{j=0}^{3} (B_{i+j} D_j + \epsilon_{B_{i+j} D_j,d}) p^2 + \sum_{j=0}^{3} \sum_{r=3}^{k-1} a_{i+j,r} p^r (D_j + \epsilon_{D_j,d}) \big) \times 10^d \Big) \bmod q \tag{4.7}$$

$$\equiv \big((R'_s + G'_s p + B'_s p^2 + \sum_{r=3}^{k-1} K_r p^r) \times 10^d \big) \bmod q \qquad \text{(by Equation 4.2)}, \tag{4.8}$$

where $R'_s = (R_s + \epsilon_{R_s})$, $G'_s = (G_s + \epsilon_{G_s})$, and $B'_s = (B_s + \epsilon_{B_s})$ are the interpolated red, green, and blue color values; R_s, G_s, and B_s are the red, green, and blue values of s when the interpolation is performed by conventional bilinear interpolation; ϵ_{R_s}, ϵ_{G_s}, and ϵ_{B_s} are the scaling errors of R_s, G_s, and B_s; and $K_r = \sum_{j=0,}^{3} (\epsilon_{D_j,d} + \epsilon_j) a_{i+j,r}\big)$ is a constant for all the shares.

The resulting scaled and cropped shadow image is then sent to the user.

4.2.3 Secret image recovery

As shown in Figure 4.2c and Figure 4.2d, in this step, the user recovers the scaled/cropped secret image from any k scaled/cropped shadow images by reconstructing the color of a pixel of the secret image from the colors of the pixels of the shadow images. The reconstruction depends on the type of operation (i,e., scaling or cropping) performed on a shadow image. The color of a pixel of a

scaled image is recovered from the colors of the scaled shadow images given by Equation 4.6 (i.e., $C'_{s,p}$'s), and the color of a pixel of a cropped image is recovered from the cropped shadow images given in Equation 4.5 (i.e., $C_{i,p}$'s).

To reconstruct the color of the s^{th} pixel in the scaled secret image, Lagrange interpolation is first applied on the color values of the s^{th} pixel of k scaled shadow images to find the polynomial

$$\begin{aligned} L(x) &= \sum_{i=0}^{k-1} C'_{s,x_i} t_i(x) \bmod q \\ &= \sum_{i=0}^{k-1} \left(\left(\left(R'_s + G'_s x_i + B'_s x_i^2 + \sum_{r=3}^{k-1} K_r x_i^r \right) \times 10^d \right) \bmod q \right) t_i(x) \bmod q \qquad (4.9) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Equation 4.6)} \\ &= \sum_{i=0}^{k-1} \left(\left(R'_s + G'_s x_i + B'_s x_i^2 + \sum_{r=3}^{k-1} K_r x_i^r \right) \times 10^d \right) t_i(x) \bmod q, \end{aligned}$$

where x_i is the i^{th} share number and $t_i(x)$ is the Lagrange basis function. By the Unisolvence theorem,

$$L(x) = \left(\left(R'_s + G'_s x + B'_s x^2 + \sum_{r=3}^{k-1} K_r x^r \right) \times 10^d \right) \bmod q.$$

Thus, we can obtain R'_s, G'_s, and B'_s by first dividing $L(x)$ by 10^d, and then solving the polynomial $\frac{L(x)}{10^d}$. Direct formulas to obtain R'_s, G'_s, and B'_s are

$$R'_s = \frac{\sum_{i=0}^{k-1} \frac{\sum_{j=0,j\neq i}^{k-1} \sum_{r=j+1,r\neq i}^{k-1} x_j x_r}{\prod_{j=0,j\neq i}^{3} (x_i - x_j)} C'_{s,x_i}}{10^d},$$

$$G'_s = -\frac{\sum_{i=0}^{k-1} \frac{\sum_{j=0,j\neq i}^{k-1} x_j}{\prod_{j=0,j\neq i}^{k-1} (x_i - x_j)} C'_{s,x_i}}{10^d},$$

and

$$B'_s = \frac{\sum_{i=0}^{k-1} \frac{1}{\prod_{j=0,j\neq i}^{k-1} (x_i - x_j)} C'_{s,x_i}}{10^d}.$$

As the magnitude of each of the scaling errors ϵ_{R_s}, ϵ_{G_s}, and ϵ_{B_s} is bound by $\pm 51 \times 10^{1-d}$ (by Equation 4.3), for a sufficiently large d, obtained red R'_s, green G'_s, and blue B'_s colors are close to R_s, G_s, and B_s, respectively.

Table 4.1: Data Sets

Name	Dimension	Size
Histo	2756×3663	28.9 MB
Drom	3000×4000	34.4 MB
Lena	512×512	768 KB
Singa	2110×1000	6.1 MB

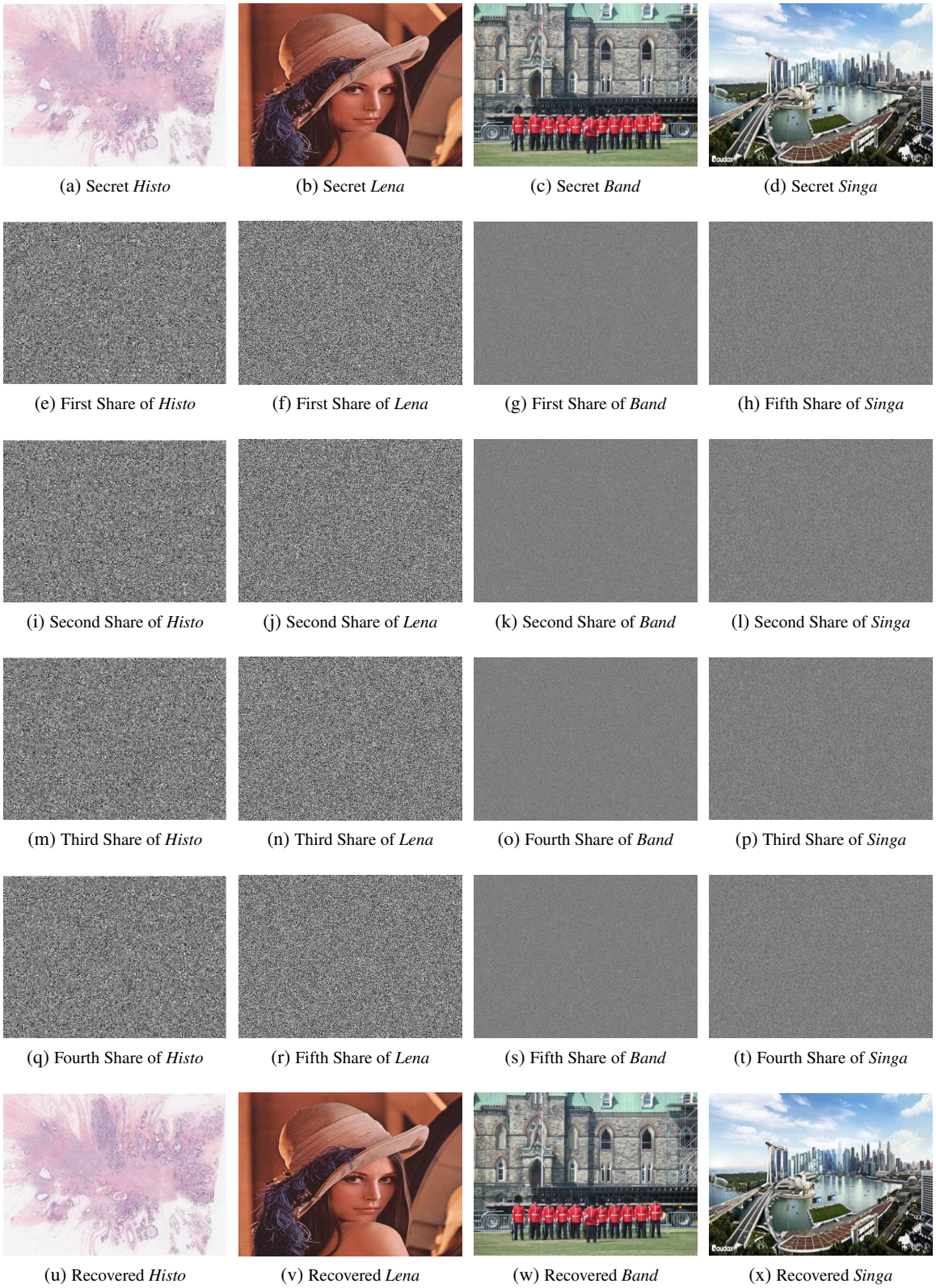
Alternatively, the color value of the s^{th} pixel in the cropped secret image is obtained by first finding the Lagrange interpolated polynomial

$$\begin{aligned} L(x) &= \sum_{i=0}^{k-1} C_{s,x_i} t_i(x) \bmod q \\ &= \sum_{i=0}^{k-1} \Big(R_s + G_s x_i + B_s x_i^2 + \sum_{j=3}^{k-1} a_j x_i^j \Big) t_i(x) \bmod q \qquad \text{(by Equation 4.5)} \\ &= (R_s + G_s x + B_s x^2 + \sum_{j=3}^{k-1} a_j x^j) \bmod q \qquad \text{(by the Unisolvence Theorem)} \end{aligned}$$

from the color values of the s^{th} pixel of k cropped shadow images; and then solving $L(x)$ to obtain R_s, G_s, and B_s. Unlike image scaling, no division by 10^d is required as the color value of a pixel of a shadow image was not multiplied by 10^d during image cropping.

4.3 Results and Analyses

We first implemented the proposed secret image sharing scheme and modified bilinear scaling using C as the programming language and on the Ubuntu platform. We used $k = 4$ and $n = 5$ for our ramp secret sharing, and $d = 2$ in modified-bilinear interpolation. We tested this experimental setup with four color (i.e., RGB) images: a histopathological image called *Histo*, a military band image called *Band*, the popular *Lena* image, and a Singapore city image called *Singa*. The details of these images are provided in Table 4.1. Furthermore, we implemented our cloud-based image scaling/cropping framework by simulating the server, datacenters, and the client in a PC powered by an *Intel Core 2 Quad 2.83 Ghz processor* and 4 GB of RAM. The simulation uses PHP to implement



(a) Secret *Histo* (b) Secret *Lena* (c) Secret *Band* (d) Secret *Singa*

(e) First Share of *Histo* (f) First Share of *Lena* (g) First Share of *Band* (h) Fifth Share of *Singa*

(i) Second Share of *Histo* (j) Second Share of *Lena* (k) Second Share of *Band* (l) Second Share of *Singa*

(m) Third Share of *Histo* (n) Third Share of *Lena* (o) Fourth Share of *Band* (p) Third Share of *Singa*

(q) Fourth Share of *Histo* (r) Fifth Share of *Lena* (s) Fifth Share of *Band* (t) Fourth Share of *Singa*

(u) Recovered *Histo* (v) Recovered *Lena* (w) Recovered *Band* (x) Recovered *Singa*

Figure 4.3: Application of $(3, 4, 5)$ randomized ramp secret sharing on images.

the datacenters, the Apache Web Server to host the datacenters, and Windows WPF to implement the client. The client-datacenter communication is facilitated by the HTTP protocol. We tested this simulation with the military band image.

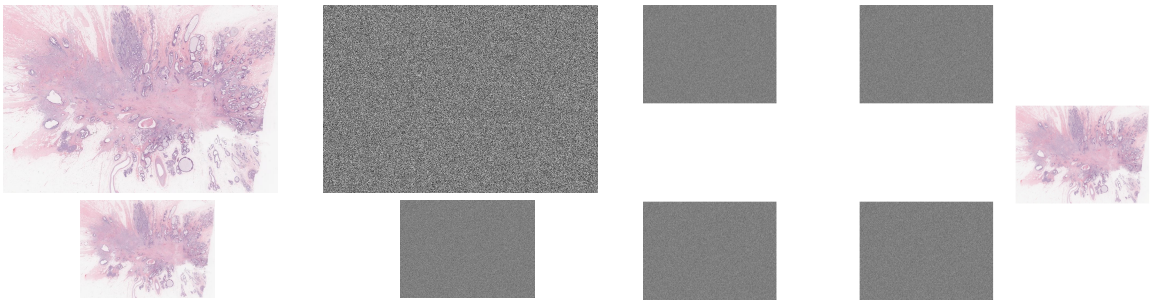
Figure 4.3 shows the results of our secret image sharing scheme. As can be verified from this figure, the shadow images are noise-like images. Therefore, they do not perceptually disclose any information of the secret image to datacenters. Figure 4.4 demonstrates scaling on shadow images, and Figure 4.6 demonstrates cropping on shadow images. As can be verified from the figures, any scaled/cropped shadow images are noise-like, but with the knowledge of at least four of them, the secret scaled/cropped image can be recovered. Figure 4.6 shows the zooming and panning operations in our secure image scaling/cropping framework.

4.3.1 Security analysis

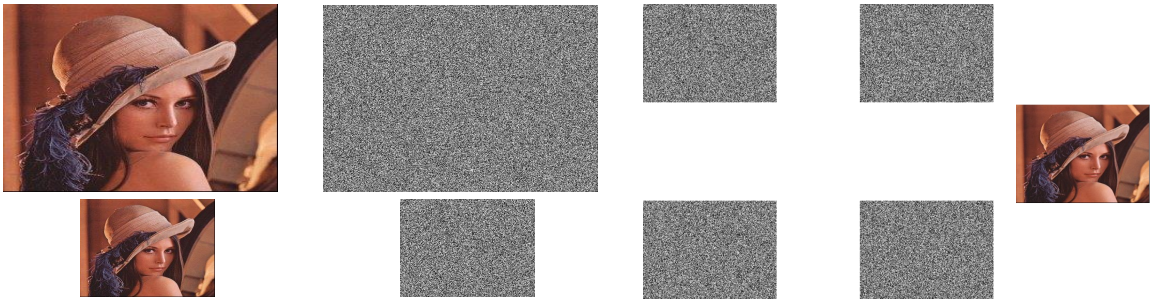
To support the claim that the proposed framework ensures data confidentiality and data integrity, we analyze it in this section.

Confidentiality

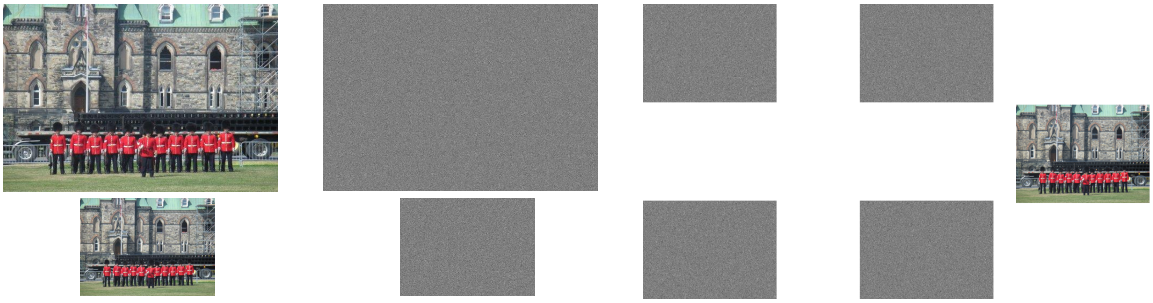
In addition to being perceptually secure, which can be verified from Figure 4.3, Figure 4.4, and Figure 4.5, our scheme is perfectly secure in a group of at most $(k-3)$ datacenters as we use 3 secrets and $k-3$ random numbers in ramp secret sharing [59]. Therefore, an adversary, irrespective of his/her computational power, cannot obtain any information about the secret image by accessing at most $(k-3)$ datacenters. Furthermore, even if the adversary is able to access $(k-2)$ or $(k-1)$ datacenters, he/she can only get $\frac{1}{3}$ or $\frac{2}{3}$ of the secret information as our secret sharing is a $(3, k, n)$ ramp secret sharing scheme [58, 59]. Although leakage of this information can decrease the number of possible values that a color component can take from 256 to $\frac{256}{3}$ or $\frac{2\times256}{3}$ respectively, it cannot change the search space of the color component from $0-255$ [58]. Therefore, the randomness in finding the secret image is less affected, and, as shown in the figures, the recovered image from $k-1$ or $k-2$ shadow images is a noise-like image.



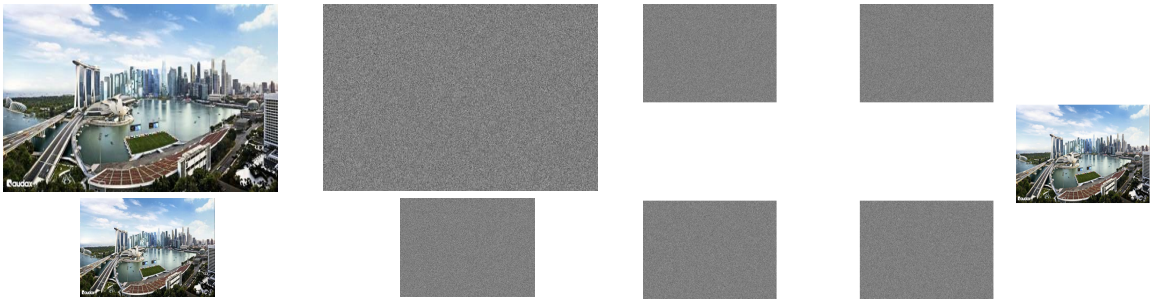
(a) Required Scaled Image (b) Scaled Shadow Image (c) Recovered Scaled Image



(d) Required Scaled Image (e) Scaled Shadow Image (f) Recovered Scaled Image

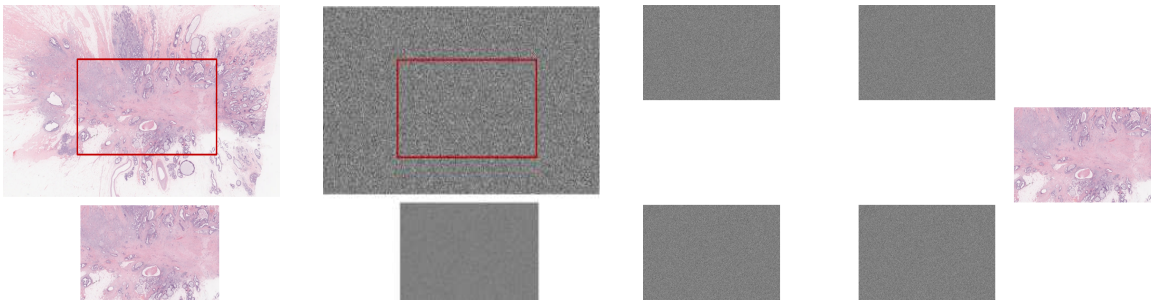


(g) Required Scaled Image (h) Scaled Shadow Image (i) Recovered Scaled Image

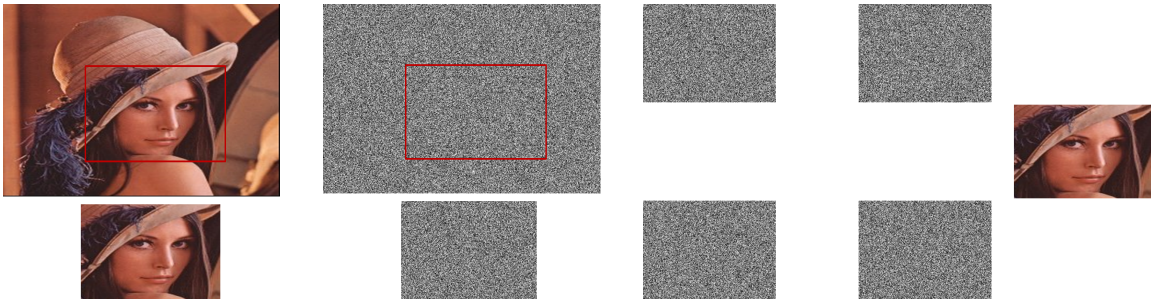


(j) Required Scaled Image (k) Scaled Shadow Image (l) Recovered Scaled Image

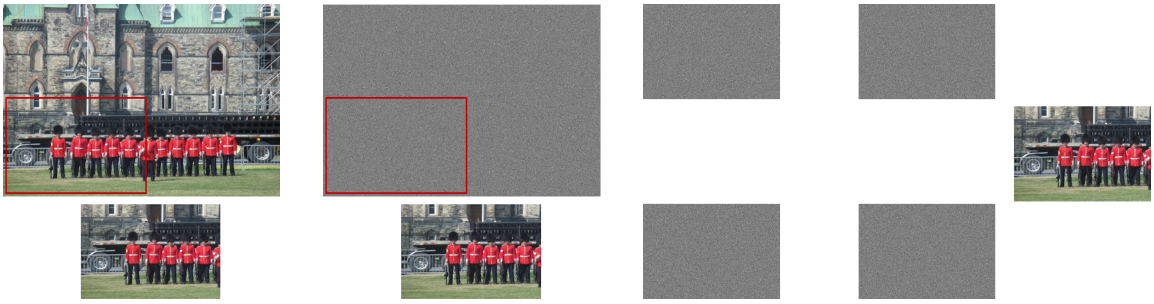
Figure 4.4: Secure cloud-based scaling of *Histo*, *Lena*, *Band*, and *Singa* images.



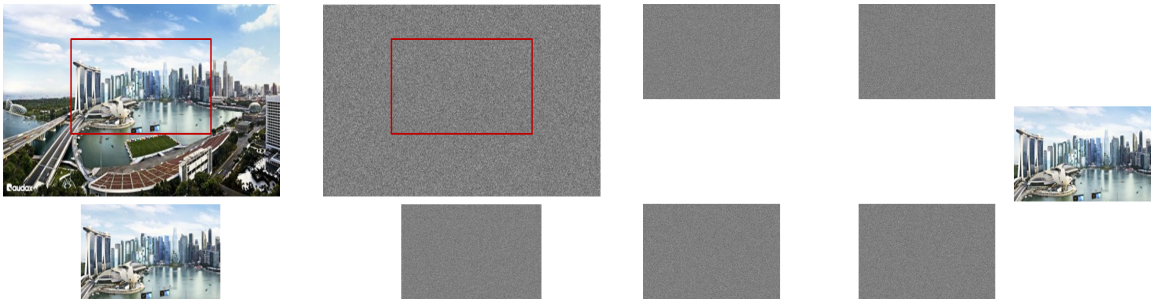
(a) Required Cropped Image (b) Cropped Shadow Image (c) Recovered Cropped Image



(d) Required Cropped Image (e) Cropped Shadow Image (f) Recovered Cropped Image



(g) Required Cropped Image (h) Cropped Shadow Image (i) Recovered Cropped Image



(j) Required Cropped Image (k) Cropped Shadow Image (l) Recovered Cropped Image

Figure 4.5: Secure cloud-based cropping of *Histo*, *Lena*, *Band*, and *Singa* images.

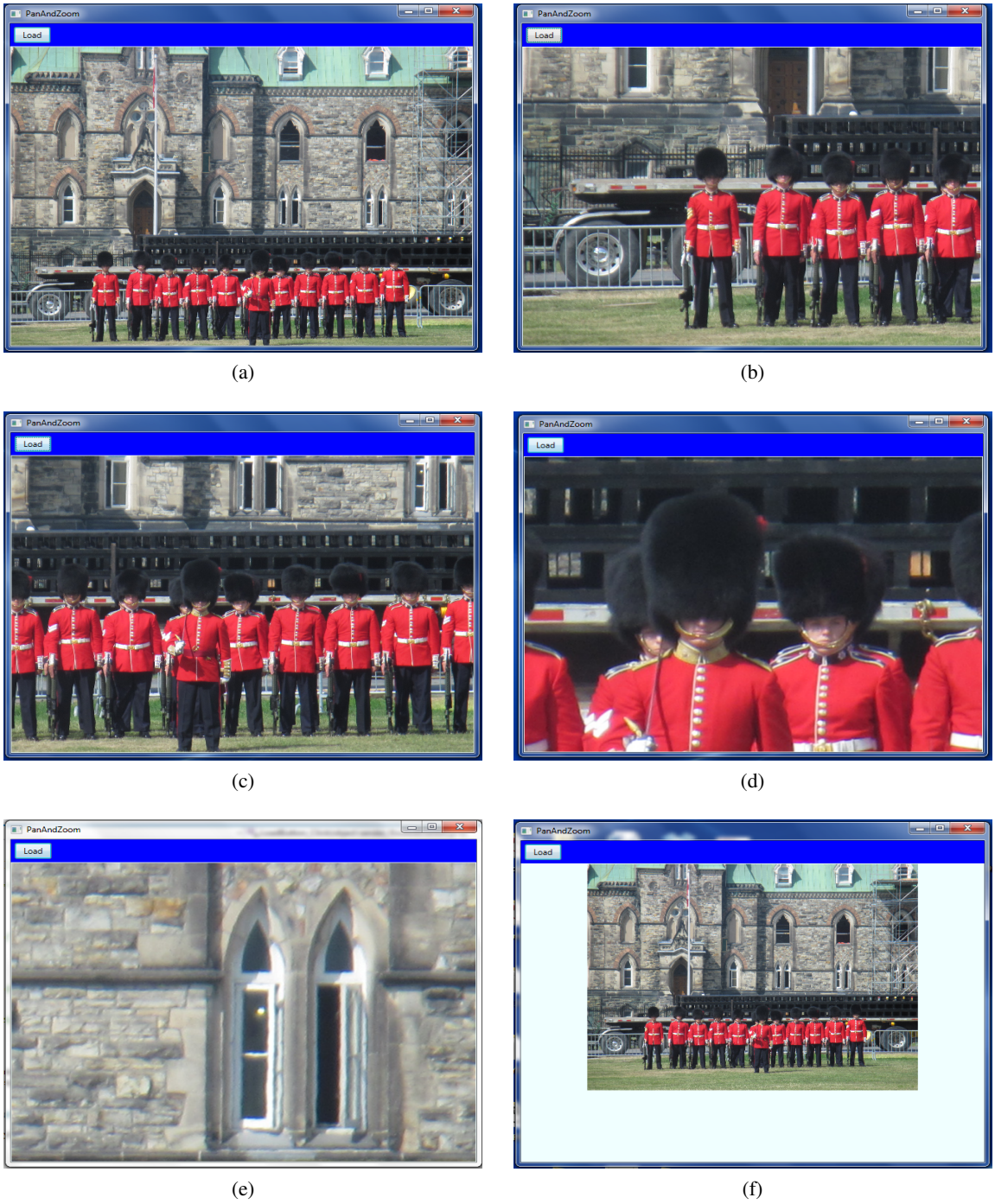
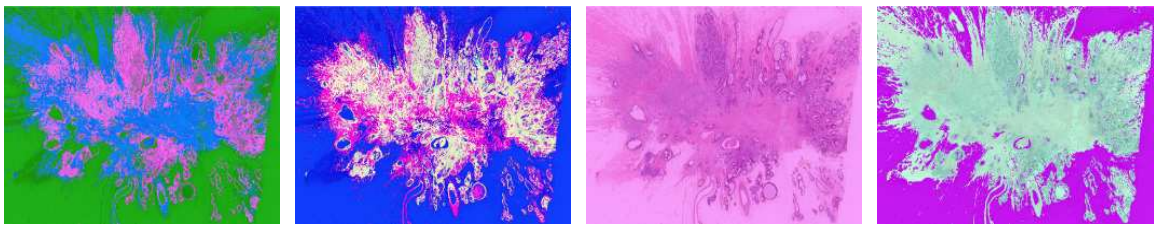


Figure 4.6: Zooming and panning operations in secure image scaling/cropping framework



(a) (b) (c) (d)

Figure 4.7: ((a), (b))- two recovered images when a shadow image of *Histo* is tampered; ((c), (d)) - two recovered images when two shadow images of *Histo* are tampered.

Integrity

By inheriting the property of secret sharing that uses the condition $k < n$, the proposed scheme also ensures data integrity by detecting any tampering with the shadow images and stopping a user from using a tampered image. The $k < n$ condition provides $\binom{n}{k}$ number of ways to recover the secret image from n shadow images. Therefore, a user can recover the same secret image using two or many ways yielding two or more recovered images. If one or more shadow images used in finding the recovered images are tampered with, then the recovered images will be dissimilar to each other and to the secret image (as shown in Figure 4.7). Therefore, by comparing the recovered images, the user can detect tampering with the shadow images and discard all the recovered images. Note that the user need to compare at most $\binom{n-1}{k} + 1$ recovered images to detect tempering.

Availability

By inheriting the property of secret sharing, our framework also ensures data availability as the client is able to reconstruct the secret image even if at most $n - k$ number of datacenters are unable to participate.

4.3.2 Performance analysis

In this section, we analyze our framework's data overhead in transmitting the shadow images to the user and computational overhead in recovering the secret image. These costs affect the latency of accessing and interacting with the stored images. We are less concerned with the computational cost of preparing the shadow images since it can be done offline.

In our scheme, a user requires the scaled/cropped shadow images of at least k datacenters to recover the scaled/cropped secret image. Therefore, if b bits are needed to represent a color value of a pixel of the shadow image, a total of bk bits are required to reconstruct the color value of a pixel of the secret image. As a result, the data overhead of our scheme is $\frac{bk-24}{24}$ times more than that of the conventional image streaming. The value of b, however, is equal to the number of bits required to represent the prime number q that is greater than $(255 + 51 \times 10^{1-d}) \times 10^d$ as $b \in$ GF(q). As a result, for a typical case of $d = 2$ and $k = 4$, the data overhead of our scheme can be calculated to be one and half times more than that of the conventional streaming – such significant data overhead is the main weakness.

The computational overhead in recovering a secret image is dependent on the computation cost w of the Lagrange interpolation that reconstructs a secret color value of a pixel, and the dimensions of the secret image. Our C implementation on a PC with an Intel Core 2 Quad 2.83 Ghz processor and 4GB of RAM recovers the secret image from the first, second, third, and fourth shadow yields $w = 0.3$ μs: due to which, approximately 78.65 ms are required to recover a secret image of 512×512 dimension.

4.4 Chapter Summary

In this chapter, we first proposed a new $(3, k, n)$ secret image sharing scheme that both preserves the homomorphic property of Shamir's secret sharing and the pixel positions of the secret image in a shadow image, to allow scaling and cropping operations on the shadow image. We then used this secret image sharing scheme to design a secure cloud-based image scaling/cropping framework that stores a shadow image in a datacenter, allows the datacenter to perform image scaling/cropping operations on its shadow image, and allows a user to recover the secret scaled/cropped image from k scaled/cropped shadow images. We showed that our scheme can also detect tampering of a shadow image, and can withstand the breakdown of certain number of datacenters.

Chapter 5

Secure Cloud-based Pre-classification Volume Ray-casting

In this chapter, we discuss our secure cloud-based pre-classification volume ray-casting framework that uses Shamir's secret sharing to hide the color information of 3D volumetric data from the datacenters.

The core idea of the proposed scheme is to use Shamir's secret sharing to hide the color information of volume data from cloud datacenters. Upon a user's request, the datacenters can then render color-hidden images, which can be used by the user to recover the secret rendered image. However, the incompatibility of the floating point operations of volume ray-casting with secret sharing is an issue.

Since we pre-compute pre ray-projection rendering components, such as gradient/normal estimation, classification, and shading, we are not concerned about their floating point operations. We are also less concerned about floating point operations associated with opacity interpolation and composition as we do not hide opacities. We are only concerned about the floating point operations of color interpolation and composition.

As discussed in Chapter 3, we can address the incompatibility of color interpolation and composition with floating point numbers by two approaches: either (i) exclude the modular prime operation from Shamir's secret sharing, or (ii) convert the floating point operation of color interpolation and

composition to fixed point operations. We use both these approaches. Using the former approach, we propose SR-MPVR (Secure Rendering by Modification of Shamir's Secret Sharing), and using the latter approach, we design SR-MSSS (Secure Rendering by Modification of Pre-classification Volume Ray-casting). As both these techniques incur high data overhead, we propose a third technique called SR-RSS (Secure Rendering by Ramp Secret Sharing), which improves upon SR-MSSS by first replacing modified Shamir's secret sharing with a modified $(3, 4, 5)$ ramp secret sharing, and then restricting the value of a share (which is a floating point number) to a smaller number and representing the restricted value with an integer.

In the following sections, we first modify pre-classification volume ray-casting to perform only fixed point operations, and then explain our secure cloud-based volume rendering framework using SR-MPVR, SR-MSSS, and SR-RSS in detail.

5.1 Pre-classification Volume Ray-casting with Fixed Point Operations

To make Shamir's secret sharing compatible with pre-classification volume ray casting, we perform the arithmetic operations involved over a finite field, in the integer domain, instead of floating point. In this section, we outline the steps that required this change and analyze the numerical precision required to bound the error in the resulting rendered color to within one.

5.1.1 Modifying interpolation

The interpolation of the colors, which is given in Equation 2.2, adds $N(s)$ multiplied values, where each multiplication is between an integer C_v and a float D_v. Therefore, to convert the interpolation to a fixed point operation, we replace D_v with a fixed point number $D_v^{(d)}$ obtained by first rounding off D_v by d decimal places and then multiplying 10^d by the rounded off value. Mathematically, $D_v^{(d)}$ is written as

$$D_v^{(d)} = \left(D_v + \epsilon_{D_v,d}\right) \times 10^d, \tag{5.1}$$

where $\epsilon_{D_v,d}$ is the round-off error.

By replacing D_v with $D_v^{(d)}$ in Equation 2.2, we obtain the scaled interpolated color as

$$
\begin{aligned}
C'_s &= \sum_{v \in N(s)} C_v D_v^{(d)} && (5.2)\\
&= \sum_{v \in N(s)} C_v \big(D_v + \epsilon_{D_v,d}\big) \times 10^d && \text{(by Equation 5.1)}\\
&= \sum_{v \in N(s)} \big(C_v D_v + C_v \epsilon_{D_v,d}\big) \times 10^d \\
&= \Big(\sum_{v \in N(s)} C_v D_v + \sum_{v \in N(s)} C_v \epsilon_{D_v,d}\Big) \times 10^d \\
&= \Big(C_s + \sum_{v \in N(s)} C_v \epsilon_{D_v,d}\Big) \times 10^d && \text{(by Equation 2.2)}\\
&= \big(C_s + \epsilon_{C_s}\big) \times 10^d, && (5.3)
\end{aligned}
$$

where $\epsilon_{C_s} = \sum_{v \in N(s)} C_v \epsilon_{D_v,d}$ is the error resulting from the interpolation step. This error satisfies the following result.

Lemma 1. *If $\epsilon_{C_s} = \sum_{v \in N(s)} C_v \epsilon_{D_v,d}$ denotes the error in the interpolation of colors, then ϵ_{C_s} is bounded by the lower bound*

$$\epsilon_{C_s,min} = -1020 \times 10^{-d}$$

and the upper bound

$$\epsilon_{C_s,max} = \begin{cases} 89.25, & \text{if } d = 1 \\ 1020 \times 10^{-d}, & \text{if } d > 1 \end{cases}.$$

Proof. It is given that $\epsilon_{C_s} = \sum_{v \in N(s)} C_v \epsilon_{D_v,d}$, where C_v is the color of a data voxel v, $\epsilon_{D_v,d}$ is the error in rounding off D_v by d decimal paces, and $N(s)$ is the number of neighbouring data voxels of the sample point s. By Inequality 2.4, C_v satisfies $0 \leq C_v \leq 255$; and by Proposition 1 and Proposition 2, $\epsilon_{D_v,d}$ satisfies $-0.5 \times 10^{-d} \leq \epsilon_{D_v,d} \leq 0.5 \times 10^{-d}$. Therefore in an ideal case, the lower bound $\epsilon_{C_s,min}$ and the upper bound $\epsilon_{C_s,max}$ of ϵ_{C_s} can be obtained by setting $\big(C_v = 255, \epsilon_{D_v,d} = -0.5 \times 10^{-d}\big)$ and $\big(C_v = 255, \epsilon_{D_v,d} = 0.5 \times 10^{-d}\big)$ respectively. However, as discussed below, for $d = 1$, we cannot choose $\epsilon_{D_v,d} = 0.5 \times 10^{-d}$ for all eight neighboring data voxels $N(s)$ of s.

By Proposition 2, the error $\epsilon_{D_v,d} = 0.5 \times 10^{-d}$ is obtained when the d^{th} digit of D_v is an odd number, the $(d+1)^{th}$ digit is five, and for all $t > 1$, the $(d+t)^{th}$ digit is zero. For $d = 1$, the lowest possible value of D_v that can satisfy this condition is 0.15. By Equation 2.5, the sum of eight D_v's corresponding to each $v \in N(s)$, however, must be equal to one. Thus, even if we choose $D_v = 0.15$, we can get a maximum of six D_v's that can result in $\epsilon_{D_v,d} = 0.5 \times 10^{-d}$. The rounding errors of the remaining two D_v's, whose sum must be $1 - 0.15 \times 6 = 0.10$, are $-\delta$ and $+\delta$, where $0.0\overline{0}1 \leq \delta \leq 0.04\overline{9}$.

However, for $d > 1$, we can get eight D_v's resulting in $\epsilon_{D_v,d} = 0.5 \times 10^{-d}$, as each of them can satisfy Proposition 2. For $d \geq 2$, one set of such D_v's can contain seven r_1's, where $r_1 = 0.00\ldots0(d-1 \text{ times})15$, and one r_2, where $r_2 = 0.99\ldots9(d-2 \text{ times})895$.

Alternatively, for $d \geq 1$, we can also get eight D_v's resulting in $\epsilon_{D_v,d} = -0.5 \times 10^{-d}$, as each of them can satisfy Proposition 1. For $d \geq 1$, one set of such D_v's can contain seven r_1, where $r_1 = 0.00\ldots0(d \text{ times})5$, and one r_2, where $r_2 = 0.99\ldots9(d-1 \text{ times})65$.

To obtain $\epsilon_{C_s,min}$, we set $C_v = 255$, $\epsilon_{D_v,d} = -0.5 \times 10^{-d}$, and $N(s) = 8$ in the formula of ϵ_{C_s}, and get

$$\begin{aligned}\epsilon_{C_s,min} &= \sum_{i=1}^{8} 255 \times (-0.5 \times 10^{-d}) \\ &= -1020 \times 10^{-d}.\end{aligned}$$

To obtain $\epsilon_{C_s,max}$, we, however, consider two cases: $d = 1$ and $d > 1$. For $d = 1$, we choose $C_v = 255$ for all six data voxels that result in round-off error $\epsilon_{D_v,d} = 0.5 \times 10^{-d}$, and for the data voxels that result in round-off error $\epsilon_{D_v,d} = +\delta$. To neutralize the negative round-off error $\epsilon_{D_v,d} = -\delta$ of the remaining data voxel, we choose its color as $C_v = 0$. Furthermore, to maximize $\epsilon_{C_s,max}$, we choose the value of δ as $0.04\overline{9}$, which can be assumed to be 0.05 with a negligible error. Therefore, for $d = 1$, we can obtain $\epsilon_{C_s,max}$ as

$$\begin{aligned}\epsilon_{C_s,max} &= \big(\sum_{i=1}^{6} 255 \times (5 \times 10^{-(1+1)})\big) + \big(255 \times 0.5\big) + \big(0 \times 0.5\big) \\ &= 89.25.\end{aligned}$$

For $d > 1$, we set $C_v = 255$, $\epsilon_{D_v,d} = 0.5 \times 10^{-d}$, and $N(s) = 8$ in the formula of ϵ_{C_s}, and obtain $\epsilon_{C_s,max}$ as

$$\begin{aligned}\epsilon_{C_s,max} &= \sum_{i=1}^{8} 255 \times (0.5 \times 10^{-d}) \\ &= 1020 \times 10^{-d}.\end{aligned}$$

As a result, we can write $\epsilon_{C_s,max}$ as

$$\epsilon_{C_s,max} = \begin{cases} 89.25, & \text{if } d = 1 \\ 1020 \times 10^{-d}, & \text{if } d > 1. \end{cases}$$

□

5.1.2 Modifying composition

The composition of color, as given in Equation 2.9, adds c multiplied values, where each multiplication is between two floating point numbers, C_{s_i} and O_i. Thus, to convert composition to integer-only operation, we replace C_{s_i} with C'_{s_i} (written in Equation 5.3), and O_i by a fixed point number

$$O_i^{(f)} = \big(O_i + \epsilon_{O_i,f}\big) \times 10^f, \tag{5.4}$$

where $\epsilon_{O_i,f}$ is the rounding error due to rounding off O_i to f decimal places.

By putting C'_s in the place of C_s and $O_i^{(f)}$ in the place of O_i in Equation 2.9, we obtain scaled composite color C' as

$$\begin{aligned} C' &= \sum_{i=1}^{c} C'_{s_i} O_i^{(f)} && (5.5) \\ &= \sum_{i=1}^{c} \big(C_{s_i} + \epsilon_{C_{s_i}}\big) O_i^{(f)} \times 10^d && \text{(by Equation 5.3)} \\ &= \sum_{i=1}^{c} \big(C_{s_i} + \epsilon_{C_{s_i}}\big)\big(O_i + \epsilon_{O_i,f}\big) \times 10^{d+f} && \text{(by Equation 5.4)} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{c} \left(C_{s_i} O_i + \epsilon_{C_{s_i}} O_i + C_{s_i} \epsilon_{O_i,f} + \epsilon_{C_{s_i}} \epsilon_{O_i,f} \right) \times 10^{d+f} \\
&= \Big(\sum_{i=1}^{c} C_{s_i} O_i + \sum_{i=1}^{c} \left(\epsilon_{C_{s_i}} O_i + C_{s_i} \epsilon_{O_i,f} + \epsilon_{C_{s_i}} \epsilon_{O_i,f} \right) \Big) \times 10^{d+f} \\
&= \Big(C + \sum_{i=1}^{c} \left(\epsilon_{C_{s_i}} O_i + C_{s_i} \epsilon_{O_i,f} + \epsilon_{C_{s_i}} \epsilon_{O_i,f} \right) \Big) \times 10^{d+f} \qquad \text{(by Equation 2.9)} \\
&= (C + \epsilon_C) \times 10^{d+f}, \qquad (5.6)
\end{aligned}
$$

where $\epsilon_C = \sum_{i=1}^{c} \left(\epsilon_{C_{s_i}} O_i + C_{s_i} \epsilon_{O_i,f} + \epsilon_{C_{s_i}} \epsilon_{O_i,f} \right)$ is the round-off error resulted in the composition step. This error's lower bound and upper bound are calculated by Theorem 2.

Remark 1. *The color C' is scaled up by 10^{d+f} in the rendering stage. Therefore, it must be scaled down by 10^{d+f} after rendering.*

Theorem 2. *If $\epsilon_C = \sum_{i=1}^{c} \left(\epsilon_{C_{s_i}} O_i + C_{s_i} \epsilon_{O_i,f} + \epsilon_{C_{s_i}} \epsilon_{O_i,f} \right)$ is the total round-off error resulting from the composition step, then ϵ_C is bounded by the lower bound*

$$
\epsilon_{C,min} = \begin{cases}
\epsilon_{C,min,1}, & \text{if } c \geq \frac{1}{0.5 \times 10^{-f}} \\
\max(\epsilon_{C,min,2}, -255), & \text{if } c < \frac{1}{0.5 \times 10^{-f}} \text{ and } c \text{ is divisible by } 4 \\
\max(\epsilon_{C,min,3}, -255), & \text{if } c < \frac{1}{0.5 \times 10^{-f}} \text{ and } c \text{ is divisible by } 2 \text{ but not divisible by } 4 \\
\max(\epsilon_{C,min,4}, -255), & \text{if } c < \frac{1}{0.5 \times 10^{-f}} \text{ and } c \text{ is not divisible by } 2
\end{cases},
$$

where

$$
\begin{aligned}
\epsilon_{C,min,1} &= -255, \\
\epsilon_{C,min,2} &= \left(510c \times 10^{-(f+d)}\right) - \left(127.5c \times 10^{-f}\right) - \left(1020 \times 10^{-d}\right), \\
\epsilon_{C,min,3} &= \left(510(c-2) \times 10^{-(f+d)}\right) - \left(127.5(c-2) \times 10^{-f}\right) - \left(1020 \times 10^{-d}\right), \\
\epsilon_{C,min,4} &= \left(510(c-1) \times 10^{-(f+d)}\right) - \left(127.5(c-1) \times 10^{-f}\right) - \left(1020 \times 10^{-d}\right);
\end{aligned}
$$

and the upper bound

$$\epsilon_{C,max} = \begin{cases} (127.5 \times \min(\frac{O_i^c}{0.5\times10^{-f}}, c) \times 10^{-f}) + 89.25O_i^c \\ \qquad + (446.2 \times \min(\frac{O_i^c}{0.5\times10^{-f}}, c) \times 10^{-(f+d)}), & \text{if } d = 1 \\ (127.5 \times \min(\frac{O_i^c}{0.5\times10^{-f}}, c) \times 10^{-f}) + (1020O_i^c \times 10^{-d}) \\ \qquad + (510 \times \min(\frac{O_i^c}{0.5\times10^{-f}}, c) \times 10^{-(f+d)}), & \text{if } d > 1, \end{cases}$$

where O_i^c satisfies the condition $255O_i^c + \epsilon_{C,max} = 255$.

Proof. It is given that $\epsilon_C = \sum_{i=1}^{c} \left(\epsilon_{C_{s_i}} O_i + C_{s_i}\epsilon_{O_i,f} + \epsilon_{C_{s_i}}\epsilon_{O_i,f}\right)$, where C_{s_i} is the interpolated color of a sample point, $\epsilon_{C_{s_i}}$ is the round-off error in color interpolation, and $\epsilon_{O_i,f}$ is the error in rounding off O_i. Of the four variables used in the formulation, C_{s_i} and $\epsilon_{C_{s_i}}$ are free from the number of sample points c, but O_i and $\epsilon_{O_i,f}$ are dependent on the value of c since O_i must satisfy $\sum_{i=1}^{c} O_i \leq 1$. Furthermore, we know that C_{s_i} and $\epsilon_{C_{s_i}}$ satisfy the conditions $0 \leq C_{s_i} \leq 255$ (by Inequality 2.6) and $\epsilon_{C_{s_i},min} \leq \epsilon_{C_{s_i}} \leq \epsilon_{C_{s_i},max}$ (by Lemma 1) respectively. Thus, the lower bound of ϵ_C, denoted by $\epsilon_{C,min}$, can be found when $C_{s_i} = 255$ and $\epsilon_{C_{s_i}} = \epsilon_{C_{s_i},min}$; and the upper bound of ϵ_C, denoted by $\epsilon_{C,max}$, can be found when $C_{s_i} = 255$ and $\epsilon_{C_{s_i}} = \epsilon_{C_{s_i},max}$. As a result, the formula to obtain the lower bound of ϵ_C, $\epsilon_{C,min}$, can be written as

$$\begin{aligned} \epsilon_{C,min} &= \epsilon_{C_s,min} \sum_{i=1}^{c} O_i + 255 \sum_{i=1}^{c} \epsilon_{O_i,f} + \epsilon_{C_s,min} \sum_{i=1}^{c} \epsilon_{O_i,f} \\ &= \epsilon_{C_s,min} O_i^c + 255 \sum_{i=1}^{c} \epsilon_{O_i,f} + \epsilon_{C_s,min} \sum_{i=1}^{c} \epsilon_{O_i,f}, \end{aligned} \tag{5.6}$$

and the formula to obtain the upper bound of ϵ_C, $\epsilon_{C,max}$, can be written as

$$\begin{aligned} \epsilon_{C,max} &= \epsilon_{C_s,max} \sum_{i=1}^{c} O_i + 255 \sum_{i=1}^{c} \epsilon_{O_i,f} + \epsilon_{C_s,max} \sum_{i=1}^{c} \epsilon_{O_i,f} \\ &= \epsilon_{C_s,max} O_i^c + 255 \sum_{i=1}^{c} \epsilon_{O_i,f} + \epsilon_{C_s,max} \sum_{i=1}^{c} \epsilon_{O_i,f}, \end{aligned} \tag{5.7}$$

where $O_i^c = \sum_{i=1}^{c} O_i$.

As $O_i^c > 0$ and $\epsilon_{O_i,f} < 0$, to obtain the value of $\epsilon_{C,min}$, we have to maximize the value of O_i^c

and minimize the value of $\sum_{i=1}^{c} \epsilon_{O_i,f}$. By Inequality 2.12, we know that $O_i^c \leq 1$; therefore, we can choose $O_i^c = 1$. In an ideal case, the minimum value of $\sum_{i=1}^{c} \epsilon_{O_i,f}$ is obtained when each $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$. As discussed below, we cannot, however, choose $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$ for all O_i's, as they must be selected such that the condition $O_i^c = 1$ is satisfied.

Let us first find the value of each $\epsilon_{O_i,f}$ when the number of sample points c satisfies $c \geq \frac{1}{0.5\times10^{-f}}$. For $c > \frac{1}{0.5\times10^{-f}}$, we can get c number of O_i's such that each O_i is less than $\frac{1}{0.5\times10^{-f}}$. When rounded off by f decimal places, each of these O_i will result in a round-off error $\epsilon_{O_i,f} = -O_i$. As a result, $\sum_{i=1}^{c} \epsilon_{O_i,f} = -1$. Similarly, for $c = \frac{1}{0.5\times10^{-f}}$, we can get c number of O_i's such that the f^{th} digit of each O_i is 0, the $(f+1)^{th}$ digit is five, and for all $t > 1$, the $(f+t)^{th}$ digit is 0. Therefore, by Proposition 1, each O_i will result in a round-off error $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$: resulting $\sum_{i=1}^{c} \epsilon_{O_i,f} = -1$.

Now let us find the value of each O_i when $c < \frac{1}{0.5\times10^{-f}}$. In this case, we can choose $c-1$ number of O_i's such that the f^{th} digit of each O_i is an even number, the $(f+1)^{th}$ digit is five, and for all $t > 1$, the $(f+t)^{th}$ digit is 0. Therefore, by Proposition 1, each O_i among $c-1$ O_i's can result in a round-off error $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$. The round-off error of the remaining O_i, however, depends on the divisibility of c with four. Firstly, if c is divisible by four, then the f^{th} digit of the remaining O_i is an even number, $(f+1)^{th}$ digit is five, and for all $t > 1$, $(f+t)^{th}$ digit is zero. Therefore, by Proposition 1, $\epsilon_{O_i,f}$ of the remaining O_i is $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$. Secondly, if c is divisible by two but not divisible by four, then the f^{th} digit of the remaining O_i is an odd number, the $(f+1)^{th}$ digit is five, and for all $t > 1$, the $(f+t)^{th}$ digit is zero. Therefore, by Proposition 2, $\epsilon_{O_i,f}$ of the remaining O_i is $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$. Finally, if c is an odd number, then the $(f+1)^{th}$ digit of each O_i is 0 and for all $t \geq 1$, the $(f+t)^{th}$ digit is 0; resulting $\epsilon_{O_i,f} = 0$.

Now, putting these values of $\epsilon_{O_i,f}$ and O_i in the equation of $\epsilon_{C,min}$, we can derive $\epsilon_{C,min}$ for four different values of c as follows.

For $c \geq \frac{1}{0.5\times10^{-f}}$, we can set $O_i^c = 1$ and $\sum_{i=1}^{c} \epsilon_{O_i,f} = -1$ in Equation 5.6, and obtain $\epsilon_{C,min}$ as

$$\begin{aligned}\epsilon_{C,min,1} &= \epsilon_{C_s,min} - 255 - \epsilon_{C_s,min} \\ &= -255.\end{aligned}$$

For $c < \frac{1}{0.5\times10^{-f}}$ and c divisible by four, we can set $O_i^c = 1$, $\epsilon_{C_s,min} = -1020 \times 10^d$, and $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$ in Equation 5.6; and obtain $\epsilon_{C,min}$ as

$$\begin{aligned}\epsilon_{C,min,2} = \big(-1020 \times 10^{-d} \times 1\big) + \big(255 \times \sum_{i=1}^{c} -0.5 \times 10^{-f}\big) \\ + \big(-1020 \times 10^{-d} \sum_{i=1}^{c} -0.5 \times 10^{-f}\big) \\ = 510c \times 10^{-(f+d)} - 127.5c \times 10^{-f} - 1020 \times 10^{-d}.\end{aligned}$$

For $c < \frac{1}{0.5\times10^{-f}}$ and c divisible by two but not divisible by four, we can set $O_i^c = 1$, $\epsilon_{C_s,min} = -1020 \times 10^{-d}$, $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$ for $c-1$ sample points, and $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$ for the remaining sample point in Equation 5.6, and obtain $\epsilon_{C,min}$ as

$$\begin{aligned}\epsilon_{C,min,3} = \big(-1020 \times 10^{-d} \times 1\big) + \big(255 \times \sum_{i=1}^{c-2} -0.5 \times 10^{-f}\big) \\ + \big(-1020 \times 10^{-d} \sum_{i=1}^{c-2} -0.5 \times 10^{-f}\big) \\ = 510(c-2) \times 10^{-(f+d)} - 127.5(c-2) \times 10^{-f} - 1020 \times 10^{-d}.\end{aligned}$$

Finally, for $c < \frac{1}{0.5\times10^{-f}}$ and c being an odd number, we can set $O_i^c = 1$, $\epsilon_{C_s,min} = -1020 \times 10^{-d}$, and $\epsilon_{O_i,f} = -0.5 \times 10^{-f}$ for $c-1$ sample points in Equation 5.6, and obtain $\epsilon_{C,min}$ as

$$\begin{aligned}\epsilon_{C,min,4} = \big(-1020 \times 10^{-d} \times 1\big) + \big(255 \times \sum_{i=1}^{c-1} -0.5 \times 10^{-f}\big) \\ + \big(-1020 \times 10^{-d} \sum_{i=1}^{c-1} -0.5 \times 10^{-f}\big) \\ = 510(c-1) \times 10^{-(f+d)} - 127.5(c-1) \times 10^{-f} - 1020 \times 10^{-d}.\end{aligned}$$

The obtained value of $\epsilon_{C,min}$ cannot be less than -255 as both the condition $0 \leq C \leq 255$ and

the condition $0 \leq C + \epsilon_{C,min} \leq 255$ must be satisfied. Therefore, we can write $\epsilon_{C,min}$ as

$$\epsilon_{C,min} = \begin{cases} \epsilon_{C,min,1}, & \text{if } c \geq \frac{1}{0.5\times10^{-f}} \\ \max(\epsilon_{C,min,2}, -255), & \text{if } c < \frac{1}{0.5\times10^{-f}} \text{ and } c \text{ is divisible by } 4 \\ \max(\epsilon_{C,min,3}, -255), & \text{if } c < \frac{1}{0.5\times10^{-f}} \text{ and } c \text{ is divisible by } 2 \text{ but not divisible by } 4 \\ \max(\epsilon_{C,min,4}, -255), & \text{if } c < \frac{1}{0.5\times10^{-f}} \text{ and } c \text{ is not divisible by } 2 \end{cases}.$$

Now let us derive the upper bound $\epsilon_{C,max}$.

We know that $O_i^c > 0$ and that $\epsilon_{O_i,f}$ can be a positive number, therefore the upper bound $\epsilon_{C,max}$, which is written in Equation 5.7, is a positive number. As a result, the obtained color $C + \epsilon_{C,max}$ can be greater than the actual color C. By Inequality 2.14, both $C + \epsilon_{C,max}$ and C, however, must satisfy the conditions $C + \epsilon_{C,max} \leq 255$ (as $0 \leq C + \epsilon \leq 255$) and $0 \leq C \leq 255$, respectively. The condition $C < 255$ is satisfied when $C_{s_i} < 255$ for at least one sample point or O_i's satisfy $O_i^c < 1$. Obtaining $\epsilon_{C,max}$ by considering the former case is tedious as it will affect the error analysis in the interpolation step. Therefore, we will consider the latter required condition: $O_i^c < 1$ that satisfies $H = 255O_i^c + \epsilon_{C,max} = 255$. In other words, to find $\epsilon_{C,max}$, we have to choose O_i's such that H is satisfied and $\sum_{i=1}^{c} \epsilon_{O_i,f}$ is maximized.

The maximum value of $\sum_{i=1}^{c} \epsilon_{O_i,f}$ is obtained when the O_i's are chosen in such a way that most O_i's have round-off error $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$. Ideally, $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$ is obtained when Proposition 2 is satisfied. Finding a fixed number of O_i's that can satisfy Proposition 2, however, is difficult as we do not know the upper bound of O_i^c in advance. Therefore, we consider the alternative case of $O_i = 5 \times 10^{-f+1} + \delta$ that can result in round-off error $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$ with $\delta \approx 0$. In this case, the number of O_i's having a round-off error $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$, however, is dependent on the number of sample points c and the value of O_i^c as discussed below.

For $c \geq \frac{O_i^c}{0.5\times10^{-f}}$, the maximum number of O_i's that can yield a round-off error 0.5×10^{-f} is approximately $\frac{O_i^c}{0.5\times10^{-f}}$, as each O_i must be equal to $0.5 \times 10^{-f} + \delta$. For $c < O_i^c$, c number of O_i's, however, can result $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$.

Therefore, by setting $C_s = 255$, and the error $\epsilon_{O_i,f} = 0.5 \times 10^{-f}$ for $\min(\frac{O_i^c}{0.5\times10^{-f}}, c)$ times in

Equation 5.7, we can obtain $\epsilon_{C,max}$ as

$$\epsilon_{C,max} = \begin{cases} (127.5 \times \min(\frac{O_i^c}{0.5\times 10^{-f}}, c) \times 10^{-f}) + 89.25 O_i^c \\ \qquad + (446.2 \times \min(\frac{O_i^c}{0.5\times 10^{-f}}, c) \times 10^{-(f+d)}), & \text{if } d = 1 \\ (127.5 \times \min(\frac{O_i^c}{0.5\times 10^{-f}}, c) \times 10^{-f}) + (1020 O_i^c \times 10^{-d}) \\ \qquad + (510 \times \min(\frac{O_i^c}{0.5\times 10^{-f}}, c) \times 10^{-(f+d)}), & \text{if } d > 1, \end{cases}$$

where O_i^c satisfies the equation $255 O_i^c + \epsilon_{C,max} = 255$. □

Due to the error in composition, C' satisfies

$$\begin{aligned} C' &\leq (C + \epsilon_{C,max}) \times 10^{d+f} \\ &\leq (255 + \epsilon_{C,max}) \times 10^{d+f} \qquad \text{(by Inequality 2.14).} \end{aligned} \tag{5.8}$$

We know that the color of a pixel is a natural number obtained by truncating the fractional part of the composited color. Therefore, a round-off error ϵ_C, which cannot change the integral part of C, does not change the rendered color, and hence the error is not effective. Thus, we claim the following.

Claim 3. *If ϵ_C is the round-off error in the composition of colors, then the effective round-off error $\epsilon_{C,eff}$ is obtained by*

$$\epsilon_{C,eff} = \lfloor C + \epsilon \rfloor - \lfloor C \rfloor,$$

where C is the composited color by conventional pre-classification volume ray-casting.

Corollary 2. *If c, the number of sample points along a ray, satisfies $c \leq 7 \times 10^t$, where $t \in \mathbb{N}$ and $t > 1$, then for $d \geq 4$ and $f \geq t + 3$, the effective rounding error $\epsilon_{C,eff}$ is bounded by ± 1.*

Proof. By Claim 3, the rounding error $\epsilon_{C,eff}$ is bounded by ± 1 when $\epsilon_{C,min}$ and $\epsilon_{C,max}$ satisfy conditions $-1 < \epsilon_{C,min} < 0$ and $0 < \epsilon_{C,max} < 1$ respectively. Therefore, we will analyze the conditions to find the value of d and f that satisfies $\epsilon_{C,min} > -1$ and $\epsilon_{C,max} < 1$.

We can write $\epsilon_{C,max}$ as $\epsilon_{C,max} = a_1 + a_2 + a_3$, where $a_1 = 127.5c \times 10^{-f}$, $a_2 = 1020 \times 10^{-d}$,

and $a_3 = 510c \times 10^{-(f+d)}$. Thus, to obtain $\epsilon_{C,max} < 1$, each of a_1, a_2, and a_3 must be less than one.

By choosing $d \geq 4$, we get $a_2 \leq 0.102$, and for this value of d, $a_1 \leq 0.8925$ when $c \leq 7 \times 10^t$ and $f \geq t + 3$. For these values of d and f, we get $a_3 \leq 0.000357$. Thus, for $d \geq 4$ and $f \geq t + 3$, $\epsilon_{C,max}$ can satisfy $\epsilon_{C,max} < 1$.

Similarly, we can write the equation of $\epsilon_{C,min}$ as $\epsilon_{C,min} = -a_1 - a_2 + a_3$. For the above chosen value of a_1, a_2, and a_3, $\epsilon_{C,min} > -1$.

Hence, we conclude that for $c \leq 7 \times 10^t$, $d \geq 4$, and $f \geq t + 3$, $\epsilon_{C,eff}$ is bounded by ± 1 □

5.2 Cloud-Based Secure Rendering

Using the above modified ray-casting operations, we can now describe how secret sharing is done. We first present our secure cloud-based rendering framework using standard Shamir's secret sharing and the modified ray-casting operations (SR-MPVR). Then, we describe SR-MSSS, which uses a weakened version of Shamir's secret sharing and standard ray-casting operations. Finally, a more efficient version that uses ramp secret sharing (SR-RSS) is presented.

5.2.1 Architecture

The architecture of our framework consists of three components: the server (e.g., a hospital) that hosts the secret data, n cloud datacenters, and the client (e.g., doctor) who intends to access the secret image (Figure 5.1). We assume that: (i) the server and the client are trusted entities, (ii) the datacenters do not share or exchange the confidential data/image with each other, and (iii) the datacenters and the client are connected to each other via a two-way high speed network.

5.2.2 SR-MPVR

As shown in Figure 5.2, the workflow of the proposed framework can be divided into four steps: (i) data preparation, (ii) ray-projection, (iii) post ray-projection rendering, and (iv) image recovery.

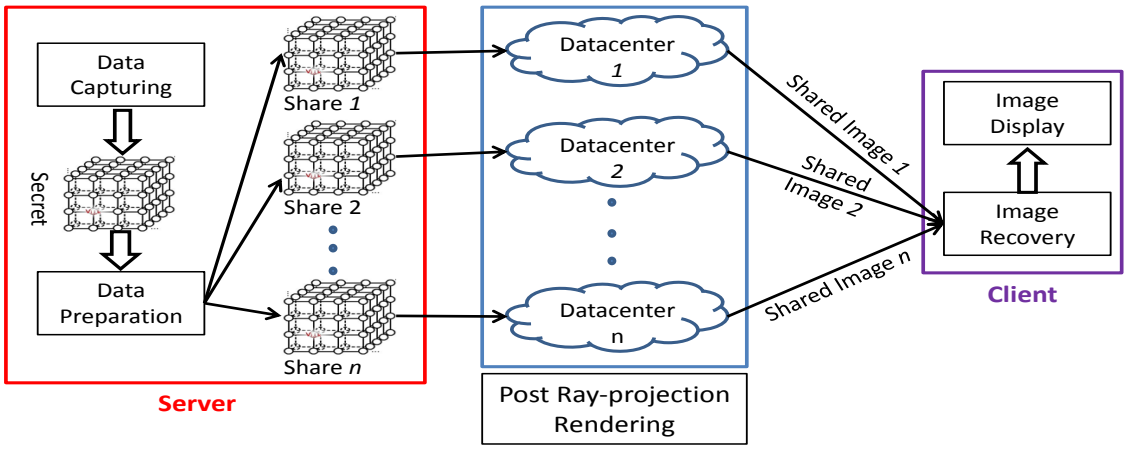


Figure 5.1: Architecture of secure cloud-based pre-classification volume ray-casting.

Data Preparation

The data preparation step creates n shares of the secret volume V. As we only hide the color information, the hidden volumes, $V_1, V_2, ...$ maintain the shape and order of the voxels of V; only the color information is modified. To achieve this, the server first finds the color and the opacity of each data voxel v of a given volume V by performing gradient estimation, classification, and shading operations; and then creates n shares of V by secret sharing the color C_v of v to n shares and copying the opacity A_v of v n times.

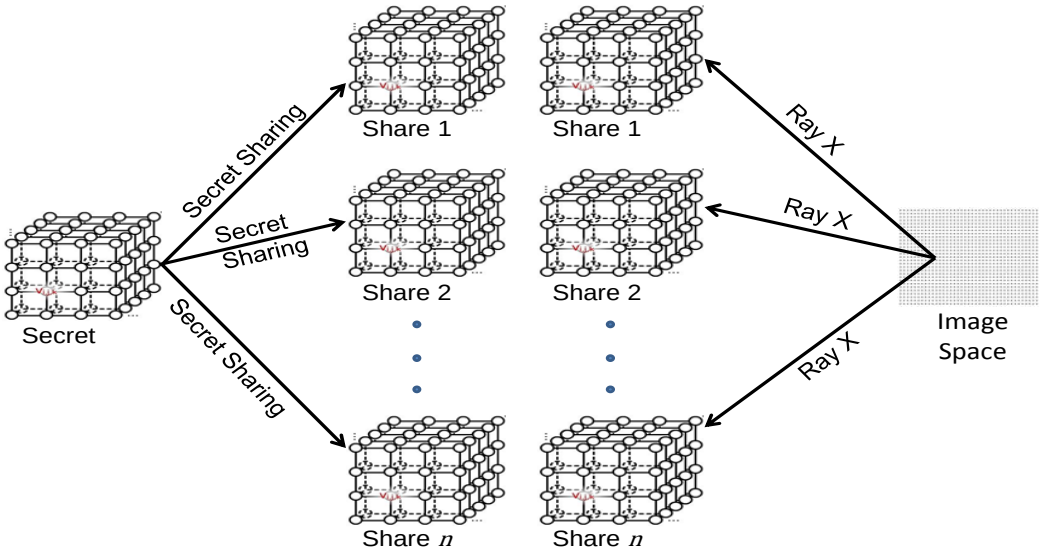
To create shares of C_v, we first choose a prime number q that is greater than the value of the maximum reconstructed secret $(255 + \epsilon_{C,max}) \times 10^{d+f}$ (which is derived in Equation 5.8), and then define a secret sharing polynomial

$$F(x) = \left(C_v + \sum_{i=1}^{k-1} a_i x^i\right) \bmod q.$$

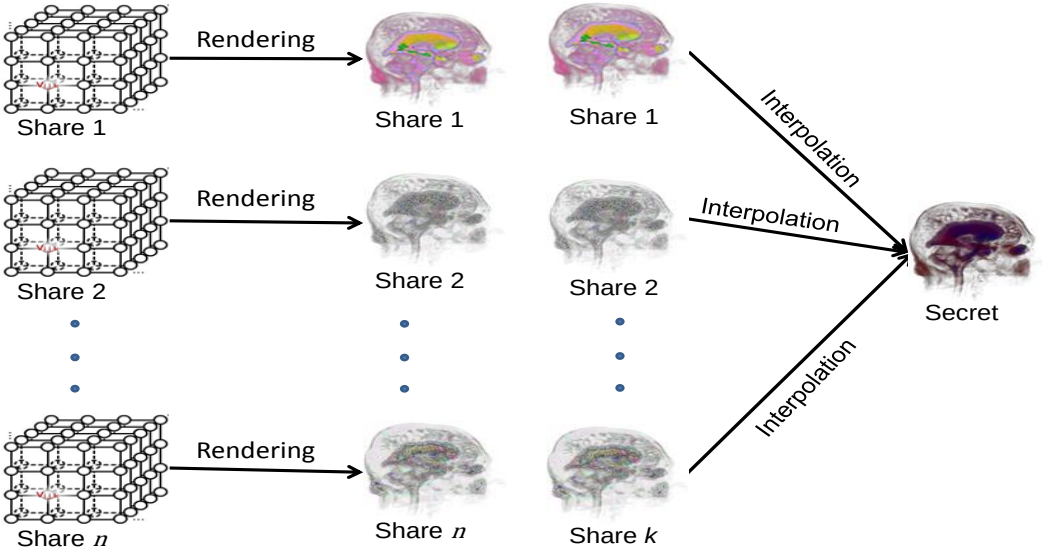
Note that we use one q to share all the C_v's as they must belong to one GF(q) in order to be added together in the future. In finding the value of q, we also fix the value of d and f in this step.

By setting $x = p$ in $F(x)$, we find the p^{th} share of C_v as

$$C_{v,p} = F(p)$$



(a) Data preparation (b) Ray-projection



(c) Post ray-projection Rendering (d) Image recovery

Figure 5.2: Workflow of secure cloud-based pre-classification volume ray-casting.

$$= \left(C_v + \sum_{i=1}^{k-1} a_i p^i\right) \bmod q$$

$$= (C_v + \alpha_p) \bmod q, \tag{5.9}$$

where

$$\alpha_p = \sum_{i=1}^{k-1} a_i p^i. \tag{5.10}$$

Next, we create the p^{th} share volume of V, V_p, and send V_p to the p^{th} datacenter.

Ray-projection

In this step, rays from the image space are projected on each share volume data. We define that the set of rays that is projected on a share volume data is equal to the set of rays that could have been projected on the secret volume data. Therefore, the set of rays that is projected on a share volume data is also equal to the set of rays that is projected on another share volume data.

Post Ray-projection Rendering

In this step, post ray-projection operations such as sampling, interpolation, and composition are performed on each share volume. The post ray-projection of one share volume is similar to others, and within a share volume, the rendering along all the projected rays is similar. We will therefore focus our further discussion on rendering along one ray in the p^{th} share volume data V_p.

Sampling: In this step, a ray projected on V_p is sampled at c sample points $s_{1,p}, s_{2,p}, \ldots, s_{c,p}$ such that xyz-coordinate $s_{i,p}$ and xyz-coordinate s_i (a sample point on the ray when it is projected on V) are the same.

Interpolation: This step finds the opacity and color of a sample point $s_p \in V_p$ by interpolating the opacities and the colors of all eight neighboring voxels of s, and the operations of this step is the same as the operation on the secret volume V.

Since we did not change the opacity, the interpolated opacity is found by the conventional opacity interpolation technique. The voxel's color, on the other hand, has changed. Therefore, color interpolation is performed by the modified-interpolation technique. The required interpolating factor is equivalent to D_v, the interpolating factor in a case of interpolation over V, as the xyz-coordinate of each sample point s and voxel v of V_p is equivalent to the xyz-coordinate of the sample point s_p and voxel v_p in V, respectively.

By putting the opacity of $v_p \in N(s_p)$ as A_v and the interpolating factor of v_p as D_v in Equation 2.3, we can get the interpolated opacity of $s_p \in V_p$ equal to A_s, which is the interpolated opacity of $s \in V$.

Similarly, by putting the color of v_p as $C_{v,p}$ and the interpolating factor of v_p as D_v in Equation 5.2, we can get the scaled interpolated color $C'_{s,p}$ as

$$
\begin{aligned}
C'_{s,p} &= \sum_{v \in N(s_p)} C_{v,p} D_v^{(d)} && (5.11)\\
&= \sum_{v \in N(s_p)} \Big(\big(C_v + \alpha_p\big) \bmod q \Big) D_v^{(d)} && \text{(by Equation 5.9)}\\
&\equiv \Big(\sum_{v \in N(s_p)} \big(C_v + \alpha_p\big) D_v^{(d)} \Big) \bmod q\\
&\equiv \Big(\sum_{v \in N(s_p)} C_v D_v^{(d)} + \sum_{v \in N(s_p)} \alpha_p D_v^{(d)} \Big) \bmod q\\
&\equiv \Big(C'_s + \alpha_p \sum_{v \in N(s_p)} D_v^{(d)} \Big) \bmod q && \text{(by Equation 5.2)}\\
&\equiv \big(C'_s + \alpha_p D_s \big) \bmod q, && (5.12)
\end{aligned}
$$

where

$$
\begin{aligned}
D_s &= \sum_{v \in N(s_p)} D_v^{(d)}\\
&= \sum_{v \in N(s_p)} \big(D_v + \epsilon_{D_v,d} \big) \times 10^d && \text{(by Equation 5.1),}
\end{aligned}
$$

is the same for all share volumes, as the interpolating factor of the q^{th} data voxel v_q of each share

volume is D_v, and each D_v is rounded off by d decimal places.

Composition: This step composites the opacities and the colors of all the sample points $s_{1,p}, s_{2,p}, \ldots, s_{c,p}$ along the projected ray. As we do not share the opacities, they can be composited by the conventional composition formula given in Equation 2.10. The composition of colors, however, must be performed by the modified-composition technique that is formulated in Equation 5.5.

As the interpolated opacity of sample point $s_{i,p}$ is A_{s_i}, by Equation 2.10 and Equation 2.11, the composited opacity of the sample points $s_{1,p}, s_{2,p}, \ldots, s_{c,p}$ is A.

The composition of color, however, uses C'_p as the color of $s_{i,p}$. Using C'_p in Equation 5.5, we obtain the scaled composited color by

$$
\begin{aligned}
C'_p &= \sum_{i=1}^{c} C'_{s_i,p} O_i^{(f)} && (5.13)\\
&\equiv \Big(\sum_{i=1}^{c} C'_{s_i,p} O_i^{(f)} \Big) \bmod q \\
&\equiv \Big(\sum_{i=1}^{c} \big(C'_{s_i} + \alpha_p D_{s_i} \big) O_i^{(f)} \Big) \bmod q && \text{(by Equation 5.12)} \\
&\equiv \Big(\sum_{i=1}^{c} C'_{s_i} O_i^{(f)} + \sum_{i=1}^{c} \alpha_p D_{s_i} O_i^{(f)} \Big) \bmod q \\
&\equiv \Big(C' + \alpha_p \sum_{i=1}^{c} D_{s_i} O_i^{(f)} \Big) \bmod q && \text{(by Equation 5.5)} \\
&\equiv \big(C' + K\alpha_p \big) \bmod q, && (5.14)
\end{aligned}
$$

where $K = \sum_{i=1}^{c} D_{s_i} O_i^{(f)}$ is the same for all the shares.

Image Recovery

Finally, an authorized user recovers the secret image from k share images obtained from k data-centers. We will show that the recovered image is close to the image rendered by conventional pre-classification volume ray-casting. As the colors and opacities of different pixels of the recovered image are found by same method, we will focus our discussion on one pixel.

As we do not hide opacities, the opacity of a pixel of a share image becomes the opacity of the

corresponding pixel of the secret image. The color of a pixel of the secret image is recovered from k shared colors (as given in Equation 5.14) by using Lagrange interpolation.

Given any k scaled composited colors: $\{C'_{x_0}, C'_{x_1}, \ldots, C'_{x_k}\}$ of k datacenters $\{x_0, x_1, \ldots, x_k\}$ (read x_i as the share number for share C'_{x_i}), let

$$\begin{aligned} y_i &= C'_{x_i} \bmod q \\ &= \left(C' + K\alpha_{x_i}\right) \bmod q && \text{(by Equation 5.14)} \\ &= \left(C' + K\sum_{j=1}^{k-1} a_j x_i^j\right) \bmod q && \text{(by Equation 5.10)} \\ &= H(x_i). \end{aligned}$$

When k of these y_i's are interpolated, they produce the Lagrange interpolated polynomial

$$L(x) = \sum_{i=0}^{k-1} y_i t_i(x),$$

where $t_i(x)$ is the Lagrange basis function. Next, suppose that

$$H(x) = \left(C' + K\sum_{j=1}^{k-1} a_j x^j\right) \bmod q$$

is a $(k-1)$-degree polynomial. Then, by the Unisolvence theorem $L(x)$ is equivalent to $H(x)$. Therefore, we can obtain C' by setting $x = 0$ in $L(x)$. Since C' is scaled up by 10^{d+f}, by Remark 1, we recover the secret color C'' by scaling down C' by 10^{d+f}. In other words,

$$\begin{aligned} C'' &= \frac{C'}{10^{d+f}} \\ &= \frac{(C + \epsilon_C) \times 10^{d+f}}{10^{d+f}} && \text{(by Equation 5.6)} \\ &= C + \epsilon_C. \end{aligned}$$

In Corollary 2, we showed that ϵ_C can be bounded by ± 1 for a sufficiently large value of d and f. Thus, we can conclude that the color rendered by our scheme (i.e, C'') is close to the color rendered

by conventional ray-casting (i.e., C).

5.2.3 SR-MSSS

We now describe the SR-MSSS method, an alternative to SR-MPVR. SR-MSSS uses floating point operations, but uses a variation of Shamir's secret sharing with a weaker security guarantee. The main workflow of SR-MSSS is similar to SR-MPVR, so we only highlight the differences below.

Data preparation

To share the colors, SR-MSSS uses the modified Shamir's secret sharing scheme. Therefore, to share the color C_v of a data voxel $v \in V$, we define a polynomial

$$F'(x) = C_v + \sum_{i=1}^{k-1} a_i x^i.$$

Using this polynomial, we find the p^{th} share of C_v as

$$\begin{aligned} C_{v,p} &= F'(p) \\ &= C_v + \alpha_p. \end{aligned} \tag{5.15}$$

Using $C_{v,p}$ as the color of the data voxel v, the p^{th} share volume is created.

Post ray-projection rendering

As we do not use any modular prime operation to share the colors of the voxels, unlike SR-MPVR, we can use the conventional color rendering algorithm to render the colors.

By putting $C_{v,p}$ as the color of $v \in V_p$ in Equation 2.2, we obtain the interpolated color $C_{s,p}$ as

$$\begin{aligned} C_{s,p} &= \sum_{v \in N(s_p)} C_{v,p} D_v \\ &= \sum_{v \in N(s_p)} \big(C_v + \alpha_p\big) D_v \qquad \text{(by Equation 5.15)} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{v \in N(s_p)} C_v D_v + \sum_{v \in N(s_p)} \alpha_p D_v \\
&= C_s + \alpha_p D_s, \qquad \text{(by Equation 2.2)}
\end{aligned} \tag{5.16}
$$

where $D_s = \sum_{v \in N(s_p)} D_v$ is a constant for all share volumes.

Next, using $C_{s,p}$ as the interpolated color of a sample point s in Equation 2.9, we obtain the composited color of c sample points $s_{1,p}, s_{2,p}, \ldots, s_{c,p}$ by

$$
\begin{aligned}
C_p &= \sum_{i=1}^{c} C_{s_i,p} O_i \\
&= \sum_{i=1}^{c} \big(C_{s_i} + \alpha_p D_{s_i}\big) O_i \qquad \text{(by Equation 5.16)} \\
&= \sum_{i=1}^{c} C_{s_i} O_i + \sum_{i=1}^{c} \alpha_p D_{s_i} O_i \\
&= C + \alpha_p \sum_{i=1}^{c} D_{s_i} O_i \qquad \text{(by Equation 2.9)} \\
&= C + K\alpha_p,
\end{aligned} \tag{5.17}
$$

where $K = \sum_{i=1}^{c} D_{s_i} O_i$ is the same for all the shares.

Image recovery

To recover the secret color from color shares (derived in Equation 5.17) obtained from k datacenters, we use Lagrange interpolation without the modular prime operation. The modified Lagrange interpolation is written as

$$
\begin{aligned}
L'(x) &= \sum_{i=0}^{k-1} C_{x_i} t_i(x), \\
&= \sum_{i=0}^{k-1} (C + K \sum_{j=1}^{k-1} a_j x^j) t_i(x). \qquad \text{(by Equation 5.10)}
\end{aligned}
$$

By the Unisolvence theorem, $L'(x) = C + K \sum_{j=1}^{k-1} a_j x^j$. Thus, by setting $x = 0$ in $L'(x)$, we can obtain the color C, which is equivalent to the color rendered by conventional ray-casting. Note

that no division operation is performed on C since the color values were not scaled in the rendering stage.

5.2.4 SR-RSS

We now present another alternative, SR-RSS, that uses ramp secret sharing to reduce the size of the share images.

Data preparation

The objective of this step is to optimize modified Shamir's secret sharing to create smaller share volumes.

We know that the three color components of a pixel, i.e., the red color R, the green color G, and the blue color B, are rendered by identical rendering operations, and that by rendering a share, a datacenter renders all the coefficients used in the secret sharing polynomial. Therefore, we use the color components R_v, G_v, and B_v of a voxel v as three secrets in the secret sharing polynomial. Although $(3, k, n)$ ramp secret sharing can reduce the data overhead by three times (as instead of creating three shares, ramp secret sharing creates only one share for all three color components), the resulting overhead is still a concern, as a shared color is represented by a floating point number, requiring 4 bytes on a typical system.

However, if we limit the values of k and n, it is possible to limit the share color to 2^{16}, thus requiring only two bytes.

To reduce the value of a color share, we choose a smaller share number at the time of secret sharing by setting the condition $k = 4$ and $n = 5$ in our ramp secret sharing. Thus the secret sharing polynomial becomes

$$F'(x) = a_0 + R_v x + G_v x^2 + B_v x^3, \tag{5.18}$$

where a_0 is a random number.

Using this polynomial and choosing a value of x smaller than five, we find the p^{th} share of all

the color components R_v, G_v, and B_v of data voxel v by

$$C_{v,p} = a_0 + R_v p + G_v p^2 + B_v p^3. \tag{5.19}$$

Using $C_{v,p}$ as the color of the data voxel v_p, we create the p^{th} share volume V_p, and then send the share volume to the corresponding datacenter. As the value of the color is less than 255 and the value of x is less than or equal to five, for $a_0 \leq 26011$, the value of $F'(x)$ cannot exceed 65536.

Note that although we can choose $n = k = 3$ to restrict the value of a color share to 3315, we do not recommend this optimization as such a scheme, by not using a random number in the secret sharing polynomial, can result in a complete breakdown of our framework. First, this $(3, 3, 3)$ ramp secret sharing does not work for a gray image as it cannot hide black color (when all color components are 0) and white color (when all color components are 255) of a voxel/pixel. Second, due to spatial coherence in an image, it is easier for an adversary to guess the color of a voxel/pixel from the known share value of the voxel/pixel and the share values of neighboring voxels/pixels of the target voxel/pixel. Similarly, we also do not recommend $n = k$ as this optimization cannot guarantee data integrity and data availability (this will be discussed in Section 5.3.1).

Post Ray-projection Rendering

Similar to SR-MSSS, we use conventional color interpolation and color composition on the color shares since no modular prime operation is required. Thus, by putting $C_{v,p}$ as the color of $v \in V_p$ in Equation 2.2, we obtain the interpolated color $C_{s,p}$ as

$$\begin{aligned}
C_{s,p} &= \sum_{v \in N(s_p)} C_{v,p} D_v \\
&= \sum_{v \in N(s_p)} \big(a_0 + R_v p + G_v p^2 + B_v p^3) D_v \qquad \text{(by Equation 5.19)} \\
&= \sum_{v \in N(s_p)} a_0 D_v + \sum_{v \in N(s_p)} R_v D_v p + \sum_{v \in N(s_p)} G_v D_v p^2 + \sum_{v \in N(s_p)} R_v D_v p^3 \\
&= D_s + R_s p + G_s p^2 + B_s p^3, \qquad \text{(by Equation 2.2)}
\end{aligned} \tag{5.20}$$

where $D_s = \sum_{v \in N(s_p)} a_0 D_v$ is a constant for all share volumes; and R_s, G_s, B_s are respectively the red color, green color, and blue color interpolated by conventional ray-casting.

Next, by using $C_{s,p}$ as the interpolated color of a sample point s in Equation 2.9, we obtain the composited color of c sample points $s_{1,p}, s_{2,p}, \ldots, s_{c,p}$ by

$$\begin{aligned} C_p &= \sum_{i=1}^{c} C_{s_i,p} O_i \\ &= \sum_{i=1}^{c} \big(D_{s_i} + R_{s_i} p + G_{s_i} p^2 + B_{s_i} p^3\big) O_i \qquad \text{(by Equation 5.20)} \\ &= \sum_{i=1}^{c} D_{s_i} O_i + \sum_{i=1}^{c} R_{s_i} O_i p + \sum_{i=1}^{c} G_{s_i} O_i p^2 + \sum_{i=1}^{c} B_{s_i} O_i p^3 \\ &= K + Rp + Gp^2 + Bp^3, \qquad \text{(by Equation 2.9)} \end{aligned} \tag{5.21}$$

where R, G, and B are the red color, green color, and blue color composited by the conventional ray-casting, and $K = \sum_{i=1}^{c} D_{s_i} O_i$ is constant for all the shares.

As the value of each $F'(p)$ is less than 65536, the value of C_p is also less than 65536. We convert C_p to a fixed point number $C_p^{(g)}$ by first rounding off C_p by g decimal places and then multiplying 10^g by the rounded off value. Mathematically, $C_p^{(g)}$ can be written as

$$C_p^{(g)} = (C_p + \epsilon_{C_p,g}) \times 10^g, \tag{5.22}$$

where $\epsilon_{C_p,g}$ is the round-off error satisfying $-0.5 \times 10^{-g} \leq \epsilon_{C_p,g} \leq 0.5 \times 10^{-g}$. We then send the scaled color share $C_p^{(g)}$ to the user.

Image Recovery

From any four share numbers (i.e., x_i's), and four color shares (i.e., $C_{x_i}^{(g)}$'s), this step finds the secret color components. To recover this secret color, we use Lagrange interpolation to obtain a polynomial

$$L'(x) = \sum_{i=0}^{3} C_{x_i}^{(g)} t_i(x)$$

$$
\begin{aligned}
&= \sum_{i=0}^{3} \Big((C_{x_i} + \epsilon_{C_{x_i},g}) \times 10^g \Big) t_i(x) \qquad \text{(by Equation 5.22)} \\
&= \sum_{i=0}^{3} (K + Rx_i + Gx_i^2 + Bx_i^3 + \epsilon_{C_{x_i},g}) t_i(x) \times 10^g \qquad \text{(by Equation 5.21)} \\
&= \Big(\sum_{i=0}^{3} (K + Rx_i + Gx_i^2 + Bx_i^3) t_i(x) + \sum_{i=0}^{3} \epsilon_{C_{x_i},g} t_i(x) \Big) \times 10^g \\
&= \big(L''(x) + \epsilon_C \big) \times 10^g, \qquad (5.23)
\end{aligned}
$$

where

$$L''(x) = \sum_{i=0}^{3} (K + Rx_i + Gx_i^2 + Bx_i^3) t_i(x),$$

and

$$\epsilon_C = \sum_{i=0}^{3} \epsilon_{C_i,g} t_i(x)$$

is the rendering error due to rounding off C_p.

Theorem 3. *If $\epsilon_C = \sum_{i=0}^{3} \epsilon_{C_{x_i},g} t_i(x)$ is the error in color composition by SR-RSS, where $\epsilon_{C_{x_i},g}$ is the error in rounding off C_{x_i} and $t_i(x)$ is the Lagrange basis function, then ϵ_C satisfies*

$$|\epsilon_C| \leq 0.5 \times 10^{-g} \times \sum_{i=2}^{5} \binom{5}{i}$$

Proof. As given, the error ϵ_C is a Lagrange interpolation of rounding errors $\epsilon_{C_{x_i},g}$. In this interpolation, each $\epsilon_{C_{x_i},g}$ satisfies $-5 \times 10^{-(g+1)} \leq \epsilon_{C_{x_i},g} \leq 5 \times 10^{-(g+1)}$, and the Lagrange basis function $t_i(x)$'s takes opposite signs for neighboring share numbers. Thus, to calculate the upper bound of ϵ_C, we consider $\epsilon_{C_{x_i},g} = 5 \times 10^{-(g+1)}$ for all positive $t_i(x)$'s, and $\epsilon_{C_{x_i},g} = -5 \times 10^{-(g+1)}$ for all negative $t_i(x)$'s. Similarly, to calculate the lower bound of ϵ_C, we consider $\epsilon_{C_{x_i},g} = -5 \times 10^{-(g+1)}$ for all positive $t_i(x)$'s, and $\epsilon_{C_{x_i},g} = 5 \times 10^{-(g+1)}$ for all negative $t_i(x)$'s.

We know that $t_i(x)$ is defined as

$$t_i(x) = \prod_{j=0, j \neq i}^{k-1} \frac{x - x_j}{x_i - x_j}.$$

In our scheme, the share number x_i satisfies $1 \leq x_i \leq 5$. Therefore, to maximize the value of

$t_i(x)$'s, we choose x_i's from the set $\{2, \ldots, 5\}$. Next, we know that $|\epsilon_C|$ will be maximized when it is associated with only one coefficient. Thus, we set $x = 0$ in $t_i(x)$.

Now by substituting the above chosen values in the formula of ϵ_C, we get

$$|\epsilon_C| \leq 0.5 \times 10^{-g} \times \sum_{i=2}^{5} \binom{5}{i}.$$

□

By the Unisolvence theorem, $L''(x)$ also can be written as

$$L'(x) = (K + Rx + Gx^2 + Bx^3 + \epsilon_C) \times 10^g.$$

Thus, we can recover the rendered colors by first dividing $L'(x)$ by 10^g, and then solving $\frac{L'(x)}{10^g}$ to obtain the second coefficient (for the red color), third coefficient (for the green color), and fourth coefficient (for the blue color) of the simplified polynomial. Using this trick, direct formulas to obtain red R', green G', and blue color B' are

$$R' = \sum_{i=0}^{3} \frac{\sum_{j=0,j\neq i}^{3} \sum_{k=j+1,k\neq i}^{3} x_j x_k}{\prod_{j=0,j\neq i}^{3}(x_i - x_j)} \frac{C_{x_i}^{(g)}}{10^g},$$

$$G' = -\sum_{i=0}^{3} \frac{\sum_{j=0,j\neq i}^{3} x_j}{\prod_{j=0,j\neq i}^{3}(x_i - x_j)} \frac{C_{x_i}^{(g)}}{10^g},$$

and

$$B' = \sum_{i=0}^{3} \frac{1}{\prod_{j=0,j\neq i}^{3}(x_i - x_j)} \frac{C_{x_i}^{(g)}}{10^g},$$

By Theorem 3, the introduced error ϵ_C satisfies

$$|\epsilon_C| \leq \frac{1}{2}\left(0.5 \times 10^{-g} \times \sum_{i=2}^{5} \binom{5}{i}\right).$$

where $R' = R + \epsilon_C$ (the same as for G' and B'). Thus, by choosing $g \geq 1$, we can obtain $|\epsilon_C| < 1$. As a result, we can conclude that the recovered color components are close to the color components rendered by conventional rendering.

Table 5.1: Data Sets

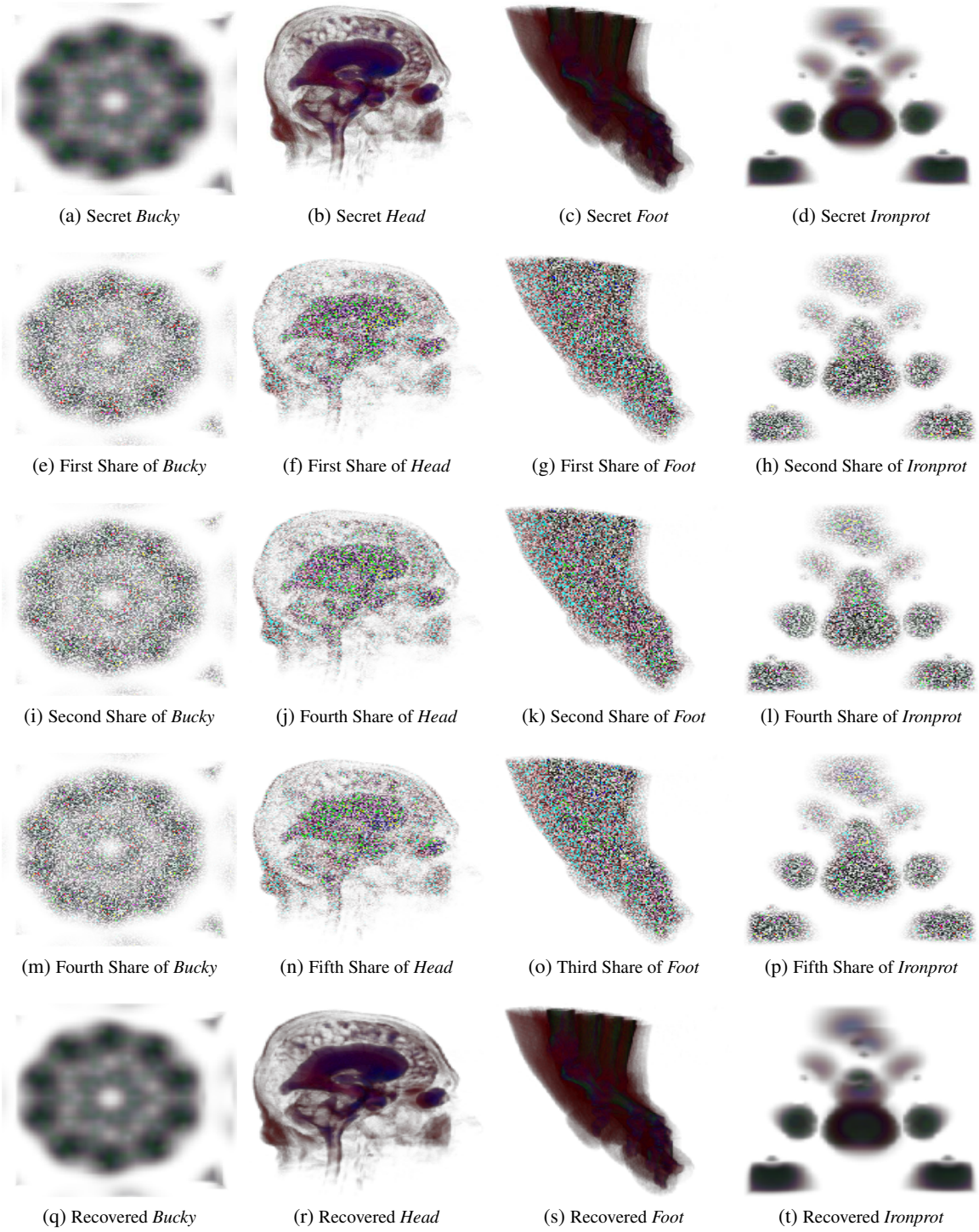
Name	Dimension	Bits per Voxel	Size
Head	$256 \times 256 \times 124$	8	7.8 MB
Foot	$256 \times 256 \times 256$	8	16 MB
Bucky	$32 \times 32 \times 32$	8	32.2 KB
Ironprot	$68 \times 68 \times 68$	8	307.3 KB

These optimizations, however, degrades security as both the use of the ramp secret sharing scheme, and the exclusion of the modular prime operation from secret sharing can disclose information about the secret. Furthermore, due to rounding error, there is a loss in information in the rendered image.

Note that we choose to optimize SR-MSSS as all of the proposed optimization tricks can be applied to SR-MSSS simultaneously. One can, however, extend the trick of using multiple secrets in a secret sharing polynomial to SR-MPVR. The trick of limiting the value of a color share by choosing a suitable share number, however, is not applicable to SR-MPVR as in the case of Shamir's secret sharing (which uses the modular prime operation), the size of a share is independent of the share number.

5.3 Results and Analyses

We simulated the server, datacenters, and the client of our framework on a PC powered by an Intel Core 2 Quad 2.83 GHz processor with 4GB of RAM. We implemented our framework by first modifying the volume ray-casting module of the open source visualization package VTK to facilitate pre-classification volume ray-casting, and then integrating secret sharing into the rendering pipeline. In the case of SR-MPVR, we used $(3, 5)$ Shamir's secret sharing in conjunction with modified pre-classification volume ray-casting; in the case of SR-MSSS, we used modified $(3, 5)$ Shamir's secret sharing in conjunction with pre-classification volume ray-casting; and in the case of SR-RSS we used $(3, 4, 5)$ modified ramp secret sharing in conjunction with pre-classification volume ray-casting. To validate our schemes, we used four sets of test volume data: *Head*, *Foot*, *Bucky*, and *Ironprot*, whose details are given in Table 5.1. As the number of sampling points along



(a) Secret *Bucky* (b) Secret *Head* (c) Secret *Foot* (d) Secret *Ironprot*

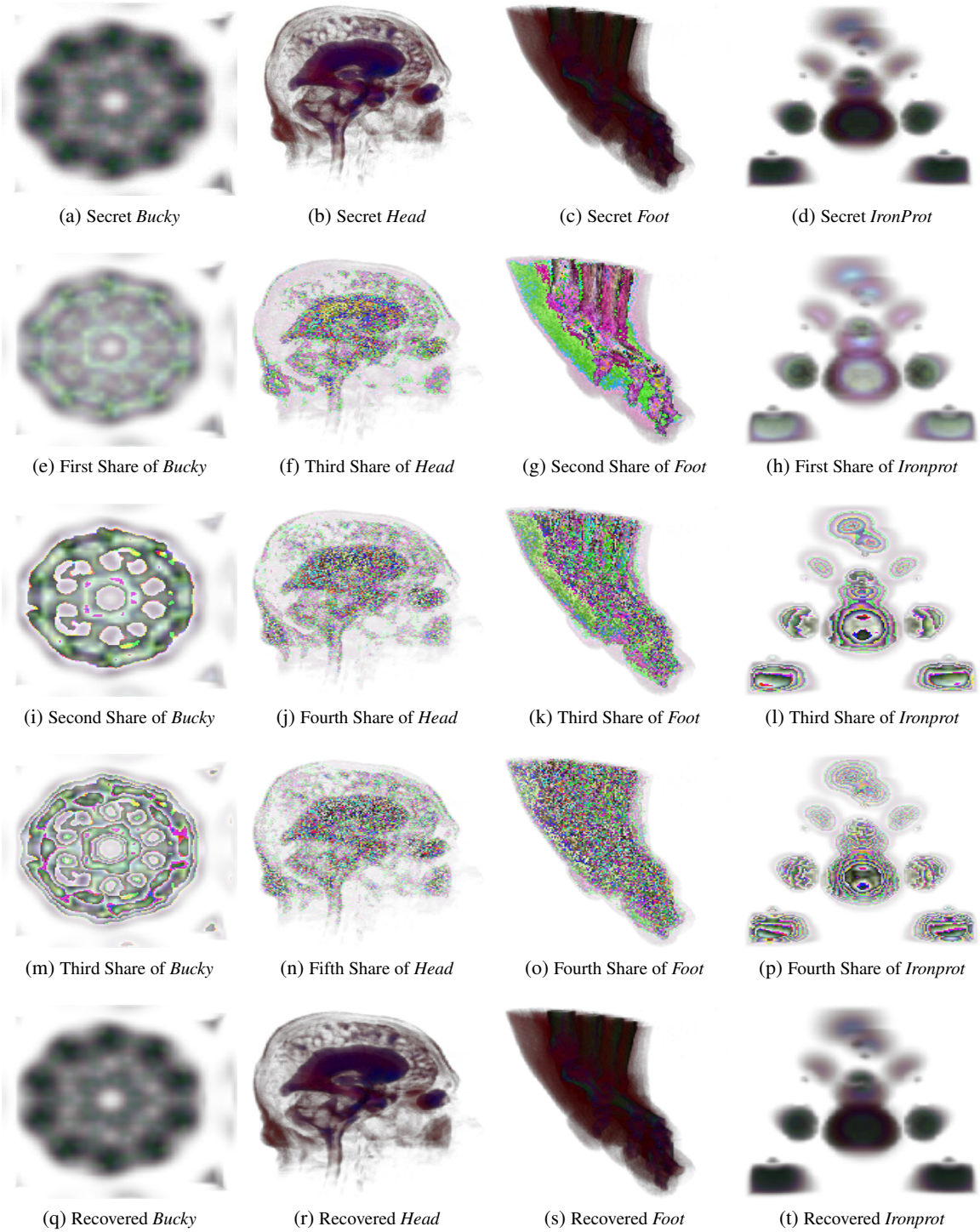
(e) First Share of *Bucky* (f) First Share of *Head* (g) First Share of *Foot* (h) Second Share of *Ironprot*

(i) Second Share of *Bucky* (j) Fourth Share of *Head* (k) Second Share of *Foot* (l) Fourth Share of *Ironprot*

(m) Fourth Share of *Bucky* (n) Fifth Share of *Head* (o) Third Share of *Foot* (p) Fifth Share of *Ironprot*

(q) Recovered *Bucky* (r) Recovered *Head* (s) Recovered *Foot* (t) Recovered *Ironprot*

Figure 5.3: Single view rendering by SR-MPVR



(a) Secret *Bucky* (b) Secret *Head* (c) Secret *Foot* (d) Secret *IronProt*

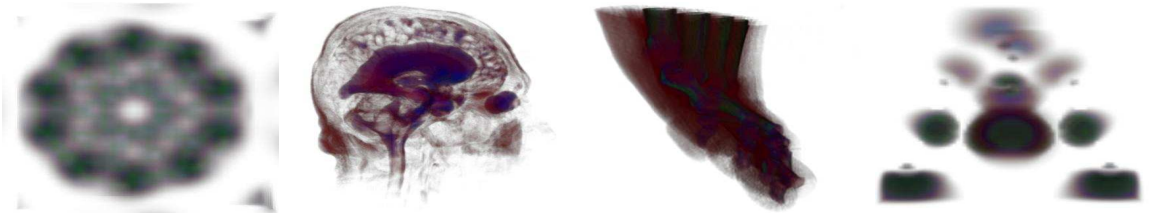
(e) First Share of *Bucky* (f) Third Share of *Head* (g) Second Share of *Foot* (h) First Share of *Ironprot*

(i) Second Share of *Bucky* (j) Fourth Share of *Head* (k) Third Share of *Foot* (l) Third Share of *Ironprot*

(m) Third Share of *Bucky* (n) Fifth Share of *Head* (o) Fourth Share of *Foot* (p) Fourth Share of *Ironprot*

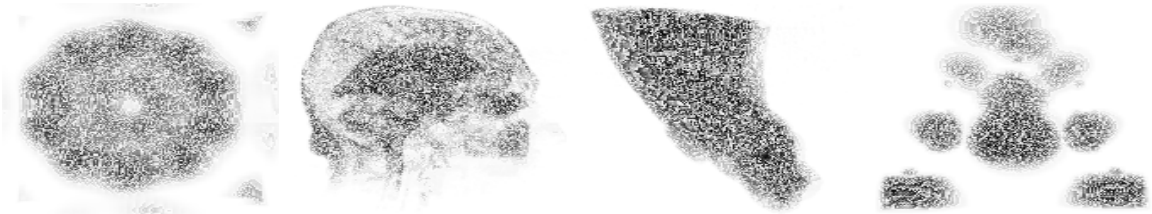
(q) Recovered *Bucky* (r) Recovered *Head* (s) Recovered *Foot* (t) Recovered *Ironprot*

Figure 5.4: Single view rendering by SR-MSSS



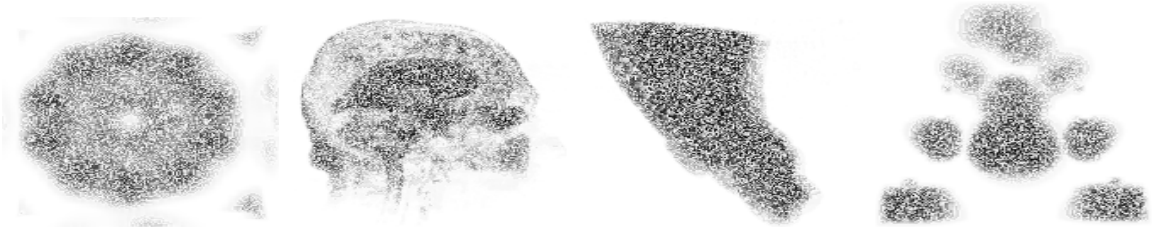
(a) Secret *Bucky* (b) Secret *Head* (c) Secret *Foot* (d) Secret *Ironprot*

(e) First Share of *Bucky* (f) First Share of *Head* (g) First Share of *Foot* (h) Second Share of *Ironprot*

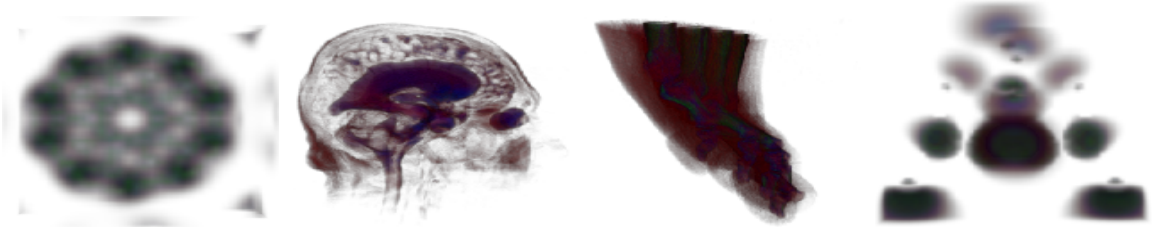


(i) Second Share of *Bucky* (j) Third Share of *Head* (k) Second Share of *Foot* (l) Third Share of *Ironprot*

(m) Third Share of *Bucky* (n) Fourth Share of *Head* (o) Third Share of *Foot* (p) Fourth Share of *Ironprot*

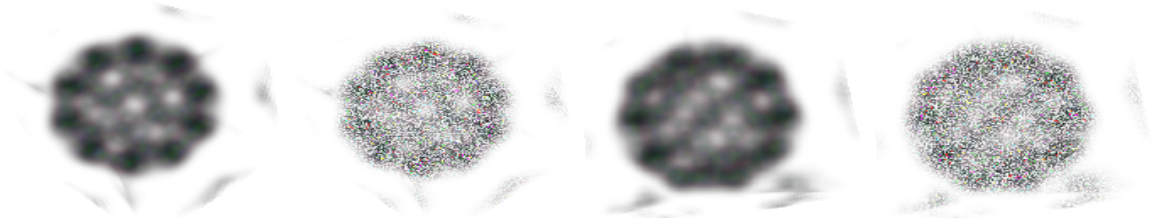


(q) Fourth Share of *Bucky* (r) Fifth Share of *Head* (s) Fourth Share of *Foot* (t) Fifth Share of *Ironprot*

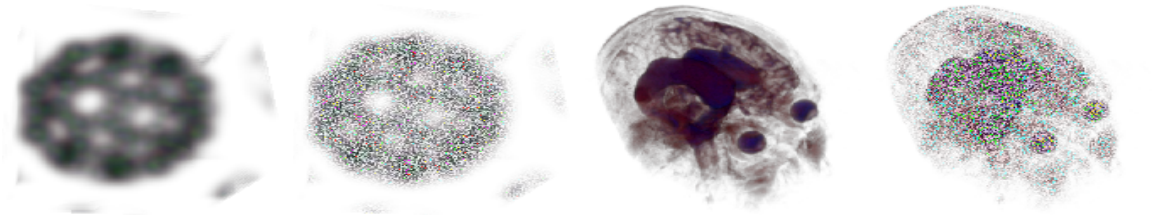


(u) Recovered *Bucky* (v) Recovered *Head* (w) Recovered *Foot* (x) Recovered *Ironprot*

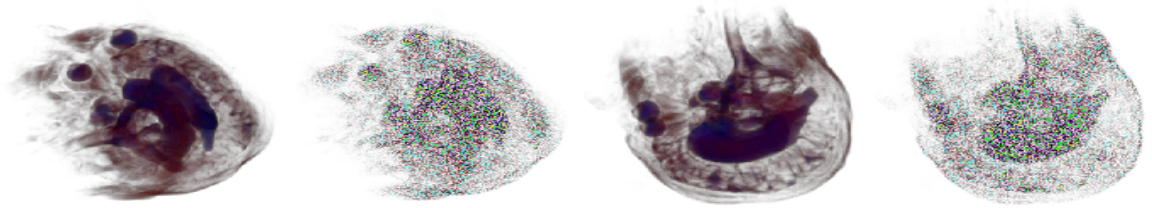
Figure 5.5: Single view rendering by SR-RSS



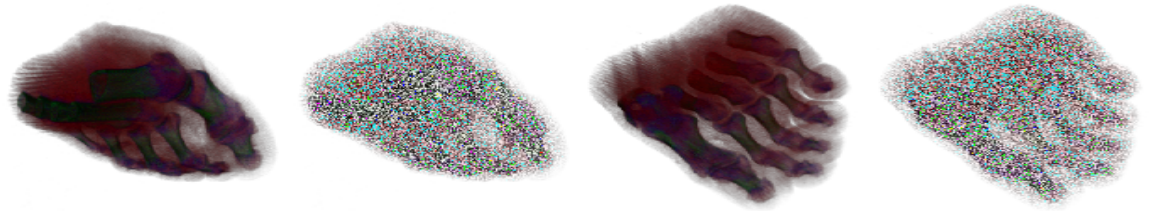
(a) Secret (b) Third Share (c) Secret (d) Third Share



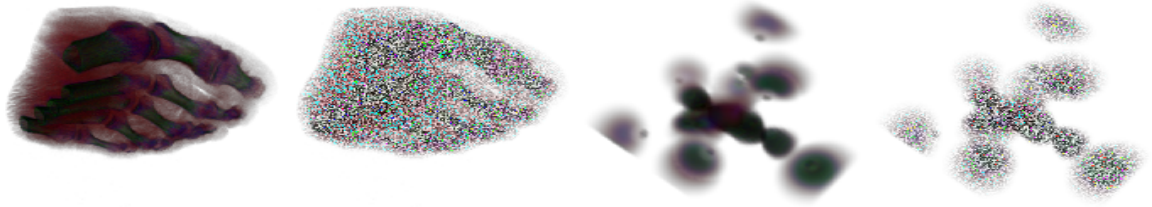
(e) Secret (f) Third Share (g) Secret (h) Third Share



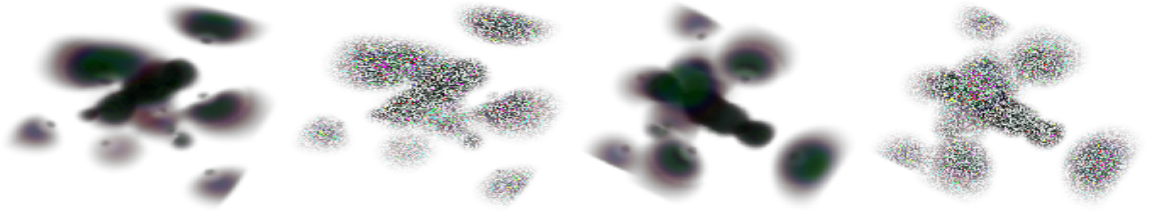
(i) Secret (j) Third Share (k) Secret (l) Third Share



(m) Secret (n) Third Share (o) Secret (p) Third Share

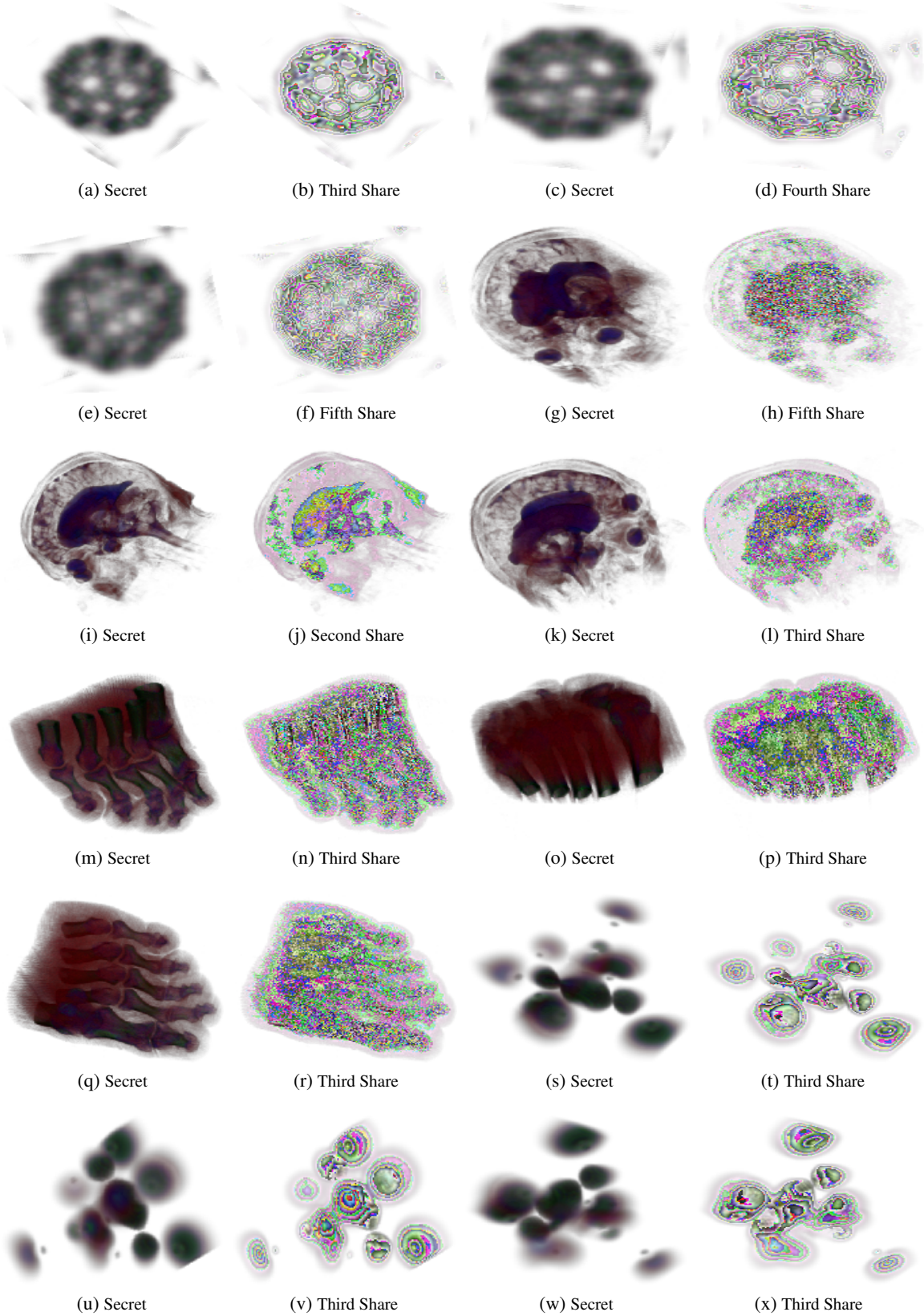


(q) Secret (r) Third Share (s) Secret (t) Third Share



(u) Secret (v) Third Share (w) Secret (x) Third Share

Figure 5.6: Multiple view rendering by SR-MPVR



(a) Secret (b) Third Share (c) Secret (d) Fourth Share

(e) Secret (f) Fifth Share (g) Secret (h) Fifth Share

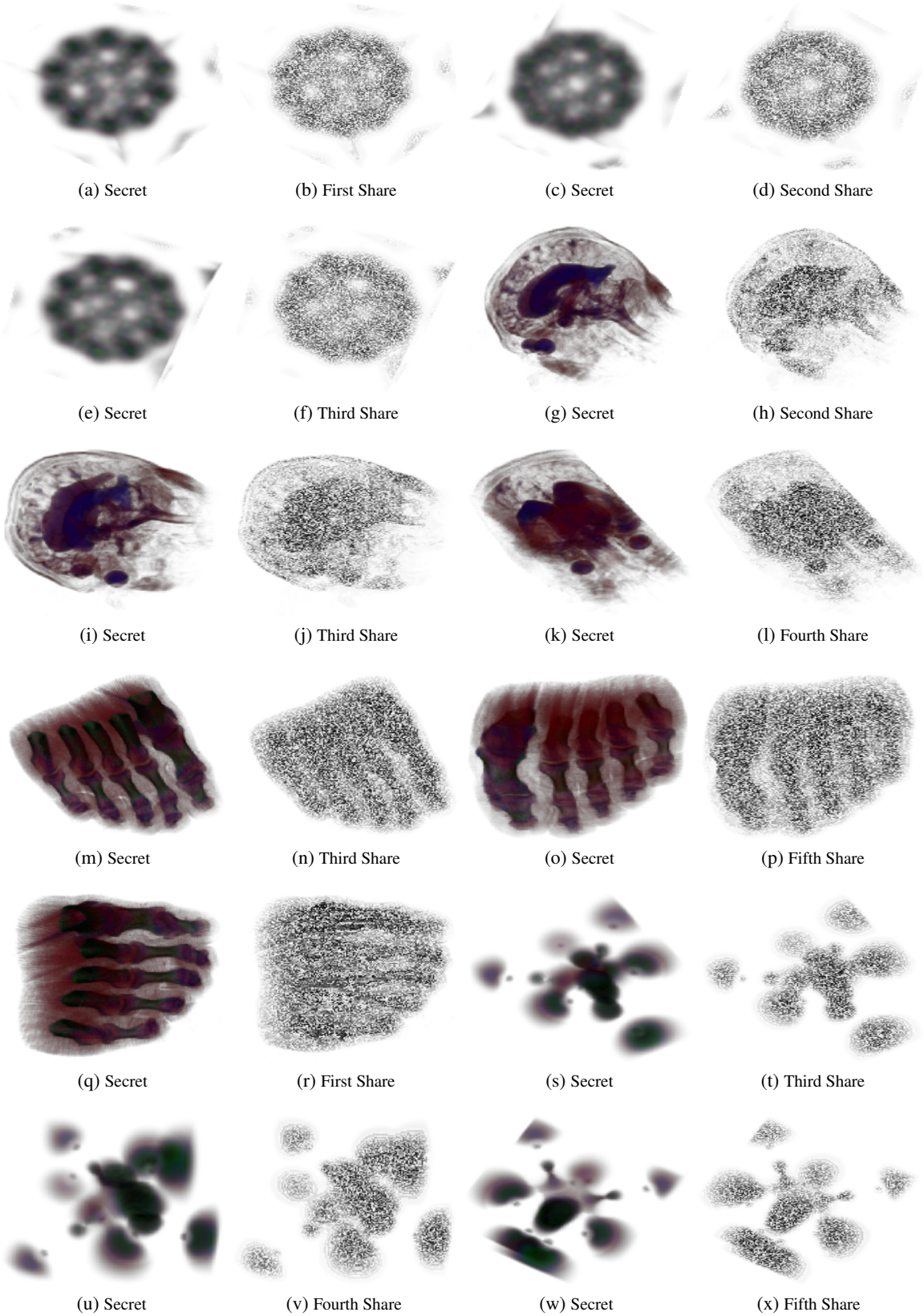
(i) Secret (j) Second Share (k) Secret (l) Third Share

(m) Secret (n) Third Share (o) Secret (p) Third Share

(q) Secret (r) Third Share (s) Secret (t) Third Share

(u) Secret (v) Third Share (w) Secret (x) Third Share

Figure 5.7: Multiple view rendering by SR-MSSS



(a) Secret (b) First Share (c) Secret (d) Second Share

(e) Secret (f) Third Share (g) Secret (h) Second Share

(i) Secret (j) Third Share (k) Secret (l) Fourth Share

(m) Secret (n) Third Share (o) Secret (p) Fifth Share

(q) Secret (r) First Share (s) Secret (t) Third Share

(u) Secret (v) Fourth Share (w) Secret (x) Fifth Share

Figure 5.8: Multiple view rendering by SR-RSS

a ray that is projected on any of these test volumes does not exceed 700, for SR-MPVR, we fixed $d = 4$ and $f = 6$ to obtain $|\epsilon| \leq 1$. For SR-RSS, we rounded off the real numbers by one decimal place (i.e., chosen $g = 1$) to keep the error below one.

Figure 5.3, Figure 5.4, and Figure 5.5 show the results of SR-MPVR, SR-MRSS, and SR-RSS, respectively. As illustrated from these figures, the color information of the secret image is hidden in share images, and the secret image can be recovered from these color-hidden images. Therefore, an adversary having access to a share image cannot perceptually infer the color coded information of the secret image. Figure 5.6, Figure 5.7, and Figure 5.8 show the results of SR-MPVR, SR-MSSS, and SR-RSS from multiple viewpoints. As can be verified from these figures, our scheme has no functional change with user's interaction with the rendered image.

Note that since the color table look-up operations are performed by the server, the proposed scheme will require server's interference to provide user interactions requiring modification of color look-up table.

5.3.1 Security analysis

In addition to perceptual security, our schemes also provide data confidentiality, data integrity, and data availability.

Confidentiality

As SR-MPVR uses Shamir's secret sharing, it is perfectly secure. Thus, an adversary, irrespective of its computation power, cannot get any information about the secret color of a voxel/pixel by accessing at most $k - 1$ datacenters.

By excluding the modular prime operation from secret sharing, SR-MSSS loses some information about the secret color in a group of less than k datacenters. As discussed in Section 3.1.1, to minimize the effect of loss of information, we can choose higher valued random numbers (i.e., a_i's) as coefficients in the secret sharing polynomial to obtain higher share values for lower share numbers. For example, even for $(2, n)$ modified secret sharing, if $a_i > 256$ and $x < 5$, the probability of knowing the secret is less than $\frac{1}{256}$. In other words, we can provide more than 256 choices to an adversary for guessing the secret, more than the number of color values possible.

By using $(3, 4, 5)$ modified ramp secret sharing, SR-RSS, in addition to losing information due to the exclusion of the modular prime operation, also loses information due to the use of multiple secrets in a secret sharing polynomial. Due to this information loss, an adversary, by accessing more than one datacenter, can easily guess some of the secret color by converting $(3, 4, 5)$ modified ramp secret sharing to $(3, 3, 5)$ modified ramp secret sharing. Therefore, SR-RSS is insecure when an adversary can access more than one datacenter. To counter such scenarios, one can easily adjust SR-RSS by introducing $(3, k + 3, n)$ ramp secret sharing, where k is the maximum number of datacenters that an adversary cannot access simultaneously.

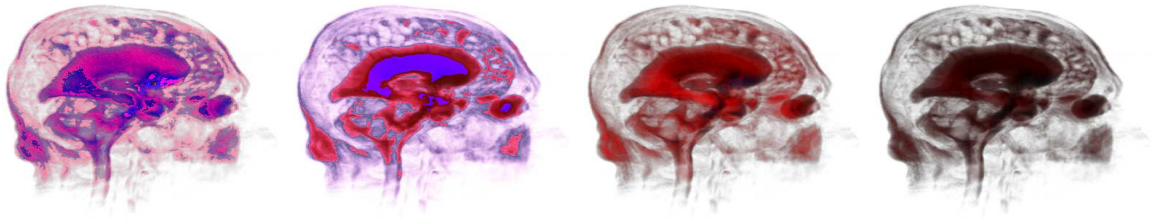
Integrity

By inheriting the property of (k, n) secret sharing, SR-MPVR, SR-MSSS, and SR-RSS ensure the integrity of data/images. The $k < n$ condition provides $\binom{n}{k}$ different ways of reconstructing the secret image. Therefore, if any adversary changes the color values of the share images (either by directly tampering with the rendered image or by tampering the share volume) of at most $n - 1$ datacenters, then the reconstructed images from the tampered share images will differ from each other, and from the secret image (as shown in Figure 5.9 for *Head*). As a result, by comparing at most $\binom{n-1}{k}$ reconstructed images, the client can detect tampering.

However, if the adversary is able to tamper with the share images of all n datacenters by obeying the homomorphic property of secret sharing, then all the tampered reconstructed images at the client site will be alike. Thus, in this case, the client will not be able to detect the tampering. Similarly, if $n = k$, then tampering with even one share image is not detectable as there can be a maximum of one recovered image. Therefore, for applications requiring data integrity, we recommend using at least one more datacenter than the number of datacenters required to recover the secret image.

Availability

By inheriting the property of (k, n) secret sharing, SR-MPVR, SR-MSSS, and SR-RSS also ensure data availability as the client is able to reconstruct the secret image even if at most $n - k$ number of datacenters are unable to participate.



(a) One Share Tampered – Non-Homomorphic (b) Two Shares Tampered – Non-Homomorphic (c) Three Shares Tampered – Homomorphic (d) No Shares Tampered

Figure 5.9: Tampering detection

5.3.2 Privacy analysis

In this section, we give the privacy analysis of SR-MPVR and SR-MSSS.

The loss in privacy of a patient is dependent on two factors: loss of identity and loss of sensitive information (e.g., the information about the diseases one is suffering from) associated with the identity. Thus, the degree of privacy loss (Γ) can be modeled as:

$$\Gamma = \Gamma_i.\Gamma_d, \tag{5.24}$$

where $0 \leq \Gamma_i \leq 1$ denotes the degree of identity loss and $0 \leq \Gamma_d \leq 1$ denotes the degree sensitive information loss.

As proposed by Saini et. al. [97], Γ_i can be further modeled as a function of explicit information I_{who} (e.g., person's name), and a set of implicit information such as I_{where} (e.g., place name), I_{what} (e.g., activity), and I_{when} (e.g., time of an activity). When any of this information is color coded, our scheme can hide it.

Irrespective of the value of Γ_i, the value of Γ can be lowered by lowering the value of Γ_d. Our scheme, by hiding the color-coded information of an image, partly lowers Γ_i. Since the shape of an image is available, the adversary can get useful information such as the type of disease that one is suffering from.

However, our scheme can preserve privacy better than the schemes that hide only the identity from the data/image (such as anonymization [98]), as hiding critical information from the third-party datacenters lowers the risk of privacy loss even when information about the identity can be known from external sources (such as from I_{who}, I_{where}, and I_{what}).

5.3.3 Performance analysis

The usability of the proposed cloud-based secured rendering technique is dependent on its computational overhead, data overhead, and the visualization latency. There are three computational overheads: first, the overhead at the server in creating n shares from the secret data voxels, second, the computational overhead at the datacenters due to use of our scheme (e.g., rendering large integers instead of rendering smaller ones), and finally, the overhead at the client in reconstructing the secret image from k share images. Similarly, there are also two types of data overheads: first, the requirement of an extra number of bits for the server to transmit n shares to n datacenters, and second, the requirement of more bits to transmit k share images to the client. The computation of shares and their distribution to datacenters are performed offline by the server. Therefore, we will not consider them in further discussion. Furthermore, we also ignore the computational overhead of a datacenter as we assume that a datacenter can render as fast as a server. However, the data overhead in transmitting k share images and the computational overhead required to reconstruct the secret image add to the latency in rendering. Therefore, we will discuss them below.

Data overhead

For SR-MPVR and SR-MSSS, our rendering framework requires k share images to reconstruct the secret image. Thus, if b number of bits are required to represent a color component of a pixel of a share image, then a total of $3bk + 8$ number of bits are required to reconstruct the color and opacity of a pixel: due to which, the data overhead to the client is $\frac{3bk-24}{32}$ times more than conventional server-side rendering. The value of b, however, is dependent on how we solve the incompatibility issue of secret sharing with ray-casting. In the case of SR-MPVR, b is dependent on the rounding off parameters d and f as the rendered color lies between 0 and q, where $q > (255 + \epsilon_{max}) \times 10^{d+f}$. For SR-MSSS, b, however, is equivalent to the number of bits required to represent a real number. Therefore, in our implementation, which uses $(3, n)$ secret sharing, 32 bits for a real number, and sets $d = 4$ and $f = 6$, SR-MPVR and SR-MSSS respectively result approximately 11 times and 8 times more data overhead to the client than the conventional server-side volume ray-casting.

In the case of SR-RSS, our rendering framework requires four color shares to recover the three

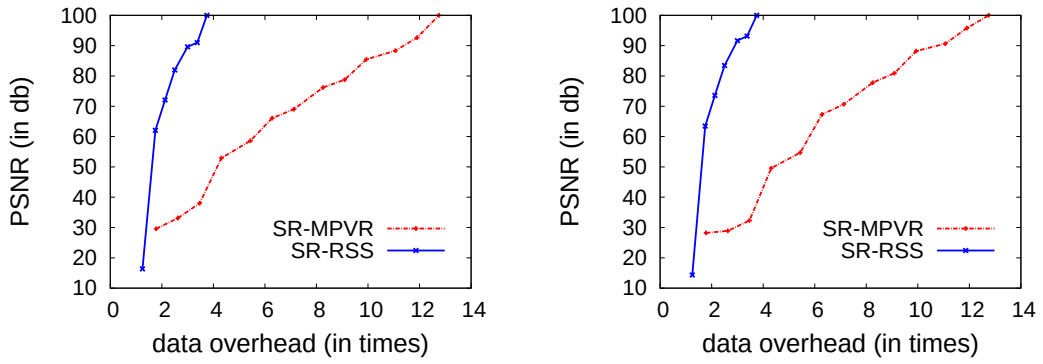
Table 5.2: Comparison among SR-MPVR, SR-MSSS, and SR-RSS

	SR-MPVR	SR-MSSS	SR-RSS
Security	Perfect security	Some information loss	More information loss than SR-MSSS
Data Overhead	Dependent on rounding bits. Typically, High	Moderate	Dependent on rounding bits. Typically, Low
Computation Cost	High	Low	Moderate
Image Quality	Lossy	Lossless	Lossy

color components of a pixel of the secret image. Thus, if b bits are required to represent a color share, then a total of $4b + 8$ bits are required by the client to obtain the secret color and opacity value of a pixel in the secret image. However, we know that the value of a color share cannot exceed 65536×10^g, where g is the number of decimal places by which a share is rounded off. Therefore, in our implementation of SR-RSS, which sets $g = 1$, a client must download approximately two times more data than the conventional server-side volume ray-casting.

Computational overhead

In our framework, the computational overhead to the client is equal to the computation cost required to recover the colors of the secret image from the share images. Therefore, client's computational overhead is dependent on the computation cost of Lagrange interpolation and the dimensions of the image. As SR-MPVR uses the modular prime operation and large integer operations, its computation cost to the client is higher than the computation costs of SR-MSSS and of SR-RSS to the client. The computation cost of SR-MSSS, however, varies with the computation cost of SR-RSS according to the required number of shares to reconstruct the secret. In our implementation, SR-MPVR and SR-MSSS take 132 ms and 29 ms, respectively, to recover a 512×512 rendered image from the first, second, and third image shares. For the same image, SR-RSS takes 34 ms to recover the secret image from the first, second, third, and fourth share images. Client's computation overhead for SR-RSS is more than that of SR-MSSS in our implementation, as the constructed polynomial for SR-RSS is one degree higher than that of SR-MSSS.



(a) *Bucky* (b) *Head*

(c) *Foot* (d) *Ironprot*

Figure 5.10: PSNR of the rendered image vs data overhead. As SR-MPVR results in a different PSNR for a fixed overhead (as for a fixed number of rounding bits $d+f$, the PSNR can change with the change in the value of d and f), we show the maximum obtained PSNR.

Image quality

By rounding off real numbers during rendering, both SR-MPVR and SR-RSS renders lossy image; but SR-MSSS, without performing rounding operations, renders lossless images.

In the cases of SP-MPVR and SR-RSS, we can, however, obtain better quality images by using higher precision fixed point numbers. As discussed in Section 5.3.3, a higher number of rounding bits can increase the data overhead, resulting in a tradeoff between the quality of the image and the data overhead. This claim can be verified from Figure 5.10, which, for our experimental setup, shows the PSNR values of Head, Foot, Bucky, and Ironprot rendered images for different overheads. As illustrated, to obtain a similar quality image, SR-MPVR leads to higher overhead than SR-RSS, as SR-MPVR rounds off two floating point numbers in contrast to only one in SR-RSS. Note that for any of these schemes, the PSNR values of different images are different for fixed overhead as the round-off error and its effect on the final rendered color are dependent on the scalar values of the data voxels.

Table 5.2 shows the security level, overheads, and image quality of all our proposed schemes: SR-MPVR, SR-MSSS, and SR-RSS. As highlighted, (i) SR-MPVR is best suited for applications prioritizing security over overheads and image quality; (ii) SR-RSS is suitable for applications requiring low overhead at cost of high security and loss of information from the rendered image; and (iii) SR-MSSS is designed for applications requiring lossless rendered images, and moderate security and overhead.

5.4 Chapter Summary

In this chapter, we proposed our secure pre-classification volume ray-casting framework that hides the color information from a 3D volumetric data from a datacenter, allows a datacenter to render a color-hidden share image from a color-hidden share volume, and allows a user to recover the secret image from the share images. We implemented the workflow of this framework both by integrating the modified Shamir's secret sharing (which do not use modular prime operation) with the conventional pre-classification volume ray-casting, and by integrating integer-only pre-classification volume ray-casting with the original Shamir's secret sharing. For the later approach, we analysed

the loss in information from a rendered image due the conversion of a floating point number to a fixed point number. For both these schemes, our analysis, however, showed that the data overhead to a user can be a concern. For applications requiring minimal overhead at the cost of high security, we proposed a third technique that uses a modified ramp secret sharing (a ramp secret sharing that does not use modular prime operation) and smaller share numbers to share the color information of the volume data. We showed that in addition to hiding confidential color information from a datacenter, all our three schemes can also ensure data integrity and data availability. None of these schemes, however, can hide the shape of an object from a datacenter.

Chapter 6

Secure Cloud-based Post-classification Volume Ray-casting

In this chapter, we propose our secure post-classification volume ray-casting scheme that hides both the color and opacity information of volume data/rendered images from cloud datacenters. In this scheme, we assume that a datacenter performs Gouard shading to render the volume data.

The core idea of the proposed scheme is to distribute the ray-casting tasks among two groups of datacenters, called Interpolator and Compositor, such that even though rendering operations other than addition and scalar multiplication are not hidden, none of the groups can know the volume data and rendered image. To hide the parts of the data/image that are added and scalar multiplied, we use Shamir's secret sharing as it is homomorphic to addition and scalar multiplication.

In this framework, a server first pre-computes pre ray-projection operations, such as gradient/normal estimation and calculation of Phong illumination factors, and of post-classification volume ray-casting, and then creates n shares of the scalar values and n shares of the outputs of the pre ray-projection operations using Shamir's (k, n) secret sharing. The shared information is then sent to Interpolator, which interpolates the shared scalars, shared gradients, and shared Phong illumination factors. Next, the Interpolator sends interpolated values to Compositor by hiding the pixel positions. Compositor completes the remaining rendering operations such as classification, shading, and composition, and sends the noise-like rendered image to a user. Finally, the user recovers the

secret image from noise-like rendered images.

The rest of this chapter is organized as follows. Section 6.1 revisits post-classification ray-casting, and points out useful observations that drove us to design the proposed framework. In Section 6.2, we propose our secure cloud-based framework, and Section 6.3 shows its results and analyses.

6.1 Post-Classification Volume Ray-Casting

As discussed in Section 2.6.2 of Chapter 2, post-classification volume ray-casting renders 3D volumetric data in a pipeline of seven independent rendering components: gradient and normal estimation, ray-projection, sampling, interpolation, classification, shading, and composition. Relevant to our work, we made the following observations about these rendering components.

- The pre ray-projection rendering operations such as gradient/normal estimation, and the calculation of Phong illumination factors Y_v (the addition of ambient reflection coefficient and diffuse reflection coefficient) and Z_v (specular reflection coefficient) are performed only per voxel once. Being independent of the user's input, these operations can be pre-computed. The rest of the rendering components, however, must be processed dynamically.

- Interpolation, which finds the scalar P_s, gradient G_s, and Phong illumination factors Y_s and Z_s of a sample point s along a projected ray, is defined as

$$P_s = \sum_{v \in N(s)} P_v D_v \tag{6.1}$$

  for P_s (the same as for G_s, Y_s, and Z_s), where $N(s)$ is the set of eight neighbouring voxels of s, P_v is the scalar value, and $0 \leq D_v \leq 1$ is the interpolating factor of the voxel $v \in N(s)$. Thus, when D_v is public, interpolation requires only additions and scalar multiplications.

- Shading, which finds the color of a sample point s, is defined as

$$C_s = C_s^{\uparrow} Y_s + Z_s, \tag{6.2}$$

  where $C_s^{\uparrow}$ is the classified color found from the color look-up table by using P_s as an index.

Thus, when C_s is public, shading requires additions and scalar multiplications.

- Composition finds the color C and opacity A along a projected ray from the shaded colors and opacities of c sample points $s_1, s_2, ..., s_c$. Color composition and opacity composition are defined as

$$C = \sum_{i=1}^{c} C_{s_i} O_i \tag{6.3}$$

and

$$A = \sum_{i=1}^{c} O_i, \tag{6.4}$$

where

$$O_i = A_{s_i} \prod_{j=i+1}^{c} \big(1 - A_{s_j}\big), \tag{6.5}$$

and A_s is the classified opacity of s. Thus, for public O_i, color composition requires additions and scalar multiplications, and opacity composition requires addition.

- After interpolation, the position and direction of a projected ray, which can disclose the coordinate of the pixel that casted it, are not required; rather, a ray must be distinguished from other rays as sample points along this ray need to be identified during color/opacity composition.

6.2 Our Framework

Based on the discussion in Section 6.1, we now design our secure rendering framework.

6.2.1 Architecture

As shown in Figure 6.1, our framework consists of four main components: the server that holds secret data, the Interpolator that contains $n \geq 2$ datacenters and performs ray-dependent rendering operations (such as sampling and interpolation), the Compositor that contains n datacenters and performs post-interpolation operations (such as classification, shading, and composition), and the client who is authorized to access the secret image.

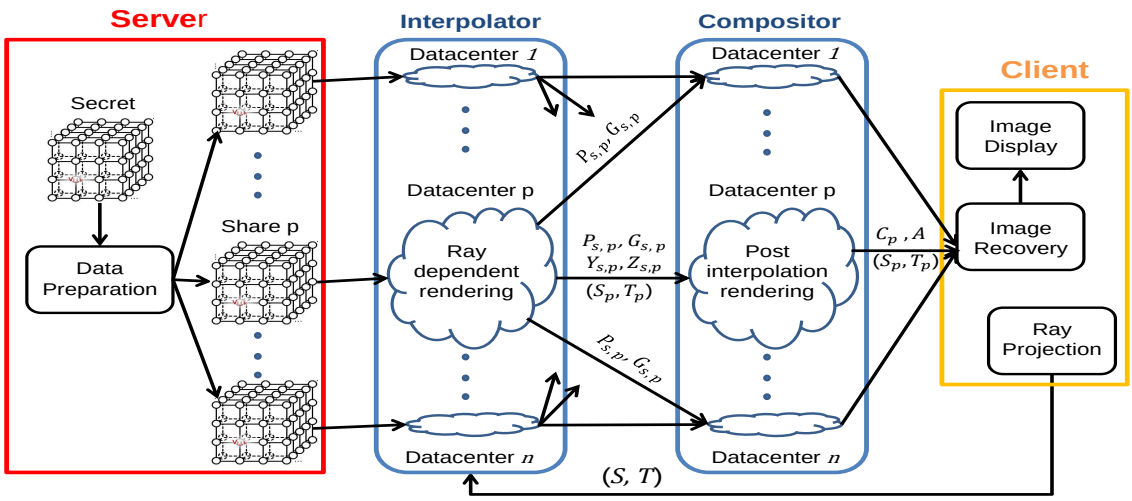


Figure 6.1: Architecture of secure cloud-based post-classification volume ray-casting

This framework assumes that (i) the server and client are the trusted entities, (ii) the datacenters are connected among themselves and with the client via a high speed network, (iii) the intra-group and inter-group communication of the datacenters is regulated, (iv) an adversary cannot access $k \leq n$ or more datacenters in Interpolator or in Compositor, and (v) an adversary cannot access a datacenter in Interpolator and Compositor simultaneously.

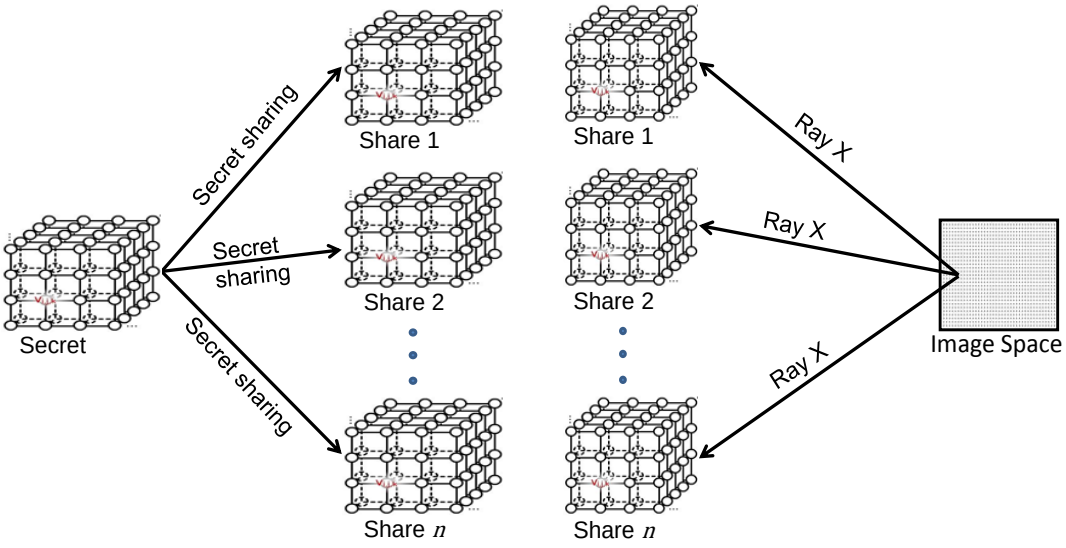
6.2.2 Workflow

As shown in Figure 6.2, the workflow of the framework can be divided into four main steps: data preparation, ray-projection, post ray-projection rendering, and image recovery. In the following, we discuss each step in detail.

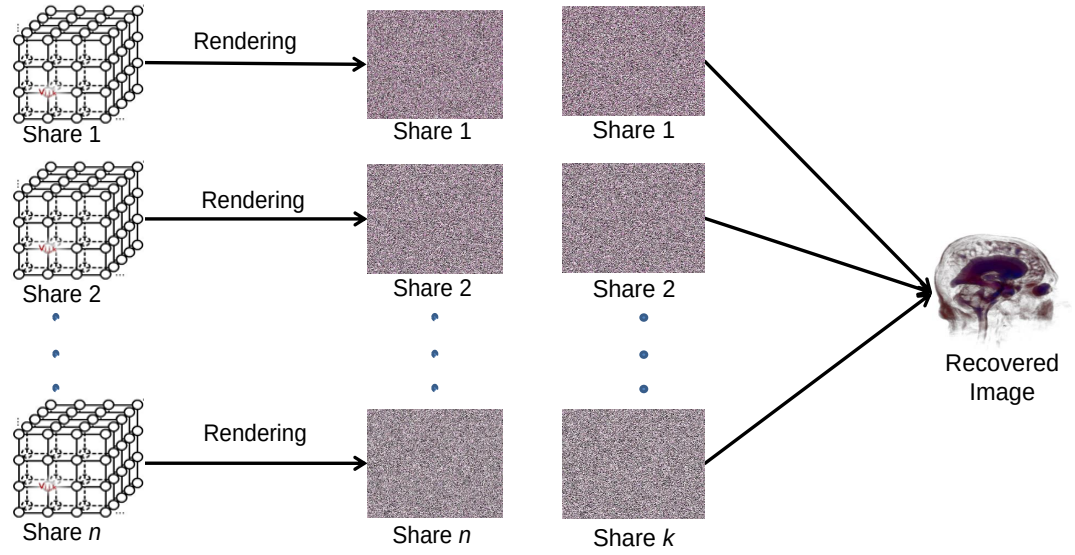
Data preparation

This step creates n shares of the secret volume V. To achieve this, the server first calculates the gradient G_v and Phong illumination factors Y_v and Z_v of each data voxel $v \in V$, and then hides the calculated values and the scalar P_v by using Shamir's (k, n) secret sharing.

However, we know that P_v, G_v, Y_v, and Z_v are floating point numbers. Therefore, they are incompatible with Shamir's secret sharing. Based on the discussion in Section 3.2, we address this issue by converting P_v, G_v, Y_v, and Z_v to fixed point numbers before secret sharing. In other words,



(a) Data preparation (b) Ray-projection



(c) Post ray-projection rendering (d) Image recovery

Figure 6.2: Workflow of secure cloud-based post-classification volume ray-casting

we represent P_v (the same as for G_v, Y_v, and Z_v) by

$$P_v^{(d)} = (P_v + \epsilon_{P_v,d}) \times 10^d, \tag{6.6}$$

where $\epsilon_{P_v,d}$ is the round-off error obtained by rounding off P_v by d decimal places.

Thereafter, we define a secret sharing polynomial

$$F(x) = \left(P_v^{(d)} + \sum_{i=1}^{k-1} a_{i,v} x^i\right) \bmod q, \tag{6.7}$$

to share $P_v^{(d)}$. Similar secret sharing polynomials are also defined for $G_v^{(d)}$, $Y_v^{(d)}$, and $Z_v^{(d)}$.

Next, by setting $x = p$ in $F(x)$, we find the p^{th} share of $P_v^{(d)}$ by

$$P_{v,p}^{(d)} = \left(P_v^{(d)} + \alpha_{v,p}\right) \bmod q, \tag{6.8}$$

where

$$\alpha_{v,p} = \sum_{i=1}^{k-1} a_{i,v} p^i. \tag{6.9}$$

Using the same technique, we also find the p^{th} share of $G_v^{(d)}$, the p^{th} share of $Y_v^{(d)}$, and the p^{th} share of $Z_v^{(d)}$ as $G_{v,p}^{(d)}$, $Y_{v,p}^{(d)}$, and $Z_{v,p}^{(d)}$, respectively.

Then, we construct the p^{th} share volume V_p using $P_{v,p}^{(d)}$ as the scalar value, $G_{v,p}^{(d)}$ as the gradient, and $Y_{v,p}^{(d)}$ and $Z_{v,p}^{(d)}$ as the Phong illumination factors of the data voxel $v_p \in V_p$ that has the same (x, y, z)-coordinate as the voxel $v \in V$. This share volume is sent to the p^{th} datacenter in Interpolator.

Ray-projection

In this step, rays from each pixel in the image space are projected on each share volume stored in a datacenter in Interpolator. As we assume that the set of rays projected on a share volume is equivalent to the set of rays that could have been projected on the secret volume, sets of rays projected to different share volumes are equivalent. We, however, do not project rays to datacenters

in Compositor.

Post ray-projection ray-casting

This operation is cooperatively performed by Interpolator and Compositor to render the volume data. Interpolator first performs the ray-dependent interpolation operations such as sampling and interpolation, and then outsources the post-interpolation operations such as classification, shading, and composition to Compositor by hiding the information about the pixel coordinates of the image space. As the post ray-projection of one share volume is similar to others, and in a share volume, the rendering along all the projected rays is similar, we will focus our further discussion on rendering along one ray on the p^{th} share volume V_p.

Ray-dependent rendering

Sampling: In this step, a ray projected on V_p is sampled at c sample points $s_{1,p}, s_{2,p}, \ldots, s_{c,p}$ such that xyz-coordinate $s_{i,p}$ and xyz-coordinate s_i (a sample point on the ray when it is projected on V) are the same.

Interpolation: To obtain the scalar value, gradient, and Phong illumination factors of a sample point s, this step interpolates the scalar values, gradients, and Phong illumination factors of eight neighbouring voxels of $s \in V_p$.

The interpolation of scalars (the same as for gradients and Phong illumination factors) involves multiplying a scalar share $P_{v,p}^{(d)}$ by floating point number D_v. Thus, we replace D_v with a fixed point number

$$D_v^{(f)} = (D_v + \epsilon_{D_v,f}) \times 10^f, \tag{6.10}$$

where $\epsilon_{D_v,f}$ is the round-off error obtained by rounding off D_v by f decimal places. As a result, the scaled interpolated scalar $P'_{s,p}$ (the same as for scaled interpolated gradient $G'_{s,p}$ and scaled

interpolated Phong illumination factors $Y'_{s,p}$ and $Z'_{s,p}$) of the sample point s of can be obtained by

$$
\begin{aligned}
P'_{s,p} &= \sum_{v \in N(s)} P^{(d)}_{v,p} D^{(f)}_{v} \\
&= \sum_{v \in N(s)} (P^{(d)}_{v} + \alpha_{v,p}) \bmod q D^{(f)}_{v} && \text{(by Equation 6.8)} \\
&= \sum_{v \in N(s)} (P^{(d)}_{v} + \alpha_{v,p}) \bmod q\big((D_v + \epsilon_{D_v,f}) \times 10^f\big) && \text{(by Equation 6.10)} \\
&\equiv \Big(\sum_{v \in N(s)} (P^{(d)}_{v} + \alpha_{v,p})(D_v + \epsilon_{D_v,f}) \times 10^f \Big) \bmod q \\
&\equiv \Big(\sum_{v \in N(s)} P^{(d)}_{v}(D_v + \epsilon_{D_v,f}) \times 10^f + \sum_{v \in N(s)} \alpha_{v,p}(D_v + \epsilon_{D_v,f}) \times 10^f \Big) \bmod q \\
&\equiv \Big(\sum_{v \in N(s)} P^{(d)}_{v}(D_v + \epsilon_{D_v,f}) \times 10^f + \alpha_{s,p} \Big) \bmod q \\
&\equiv \Big(\sum_{v \in N(s)} (P_v + \epsilon_{P_v,d}) \times 10^d \times (D_v + \epsilon_{D_v,f}) \times 10^f + \alpha_{s,p} \Big) \bmod q && \text{(by Equation 6.6)} \\
&\equiv \Big(\sum_{v \in N(s)} (P_v D_v + \epsilon_{P_v,d} D_v + P_v \epsilon_{D_v,f} + \epsilon_{P_v,d} \epsilon_{D_v,f}) \times 10^{d+f} + \alpha_{s,p} \Big) \bmod q \\
&\equiv \big((P_s + \epsilon_{P_s}) \times 10^{d+f} + \alpha_{s,p} \big) \bmod q, && \text{(by Equation 6.1)} \qquad (6.11)
\end{aligned}
$$

where P_s is the scalar value interpolated by conventional rendering,

$$\epsilon_{P_s} = \sum_{v \in N(s)} (\epsilon_{P_v,d} D_v + \epsilon_{D_v,f} P_v + \epsilon_{P_v,d} \epsilon_{D_v,f})$$

is the error during interpolation, and

$$\alpha_{s,p} = \sum_{v \in N(s)} (D_v + \epsilon_{D_v,f}) \alpha_{v,p} \times 10^f. \qquad (6.12)$$

Note that as the interpolated scalars and gradients are in shared form, by using these values in look up tables, a datacenter cannot get the secret colors and opacities.

To complete the remaining rendering operations, the p^{th} datacenter in Interpolator copies its interpolated scalar $P'_{s,p}$ and interpolated gradient $G'_{s,p}$ to each datacenter in Compositor, but sends

the interpolated Phong illumination factors $Y'_{s,p}$ and $Z'_{s,p}$ only to the p^{th} datacenter of Compositor.

To hide the pixel positions of the image space from Compositor while distinguishing the sample points along a ray from other sample points, the p^{th} datacenter creates a proxy identification $PID_{(S,T)}$ of a ray that originated from a pixel having the coordinates (S, T), and in place of (S, T), sends this proxy to the p^{th} datacenter of Compositor. The secret (S, T), however, is required during the image recovery stage. Thus, the p^{th} datacenter of Interpolator also sends the p^{th} share of (S, T) to the p^{th} datacenter of Compositor. If (S_p, T_p) denotes the p^{th} share of (S, T), then by Shamir's secret sharing S_p (the same as for T_p) can be found by

$$S_p = (S + \sum_{i=1}^{k-1} a_i x^i) \bmod q', \tag{6.13}$$

where a_i is a random number and $q' > S$ is a prime number.

Post-interpolation rendering

In this step, each datacenter in Compositor performs post-interpolation rendering operations, such as classification, shading, and composition, on the interpolated data received from the datacenters of Interpolator. As these operations are the same in all the datacenters, we focus our discussion on the p^{th} datacenter.

Classification: The objective of this step is to find the classified color and classified opacity of a sample point using scalar shares and gradient shares. The p^{th} datacenter of Compositor receives n shares of scalars and gradients from the datacenters of Interpolator. Thus, using Lagrange interpolation, we can recover the secret interpolated scalars and gradients. Mathematically, the Lagrange interpolated polynomial for a scalar share (the same as for gradient) can be found by

$$\begin{aligned} L(x) &= \sum_{i=0}^{k-1} P'_{s,p} t_i(x) \bmod q \\ &= \sum_{i=0}^{k-1} \Big(\big((P_s + \epsilon_{P_s}) \times 10^{d+f} + \alpha_{s,p} \big) \bmod q \Big) t_i(x) \bmod q \qquad \text{(by Equation 6.11)} \end{aligned}$$

$$= \sum_{i=0}^{k-1} (P''_s \times 10^{d+f} + \alpha_{s,p}) t_i(x) \bmod q,$$

$$= \sum_{i=0}^{k-1} \big(P''_s \times 10^{d+f} + \sum_{v \in N(s)} (D_v + \epsilon_{D_v,f}) \alpha_{v,p} \times 10^f\big) t_i(x) \bmod q \tag{6.14}$$

(by Equation 6.12)

$$= \sum_{i=0}^{k-1} \Big(P''_s \times 10^{d+f} + \sum_{v \in N(s)} (D_v + \epsilon_{D_v,f}) \big(\sum_{i=1}^{k-1} a_{i,v} p^i\big) \times 10^f\Big) t_i(x) \bmod q \tag{6.15}$$

(by Equation 6.9)

$$= \sum_{i=0}^{k-1} \Big(P''_s \times 10^{d+f} + \sum_{i=1}^{k-1} \beta_{s,p} p^i\Big) t_i(x) \bmod q, \tag{6.16}$$

where $t_i(x)$ is the Lagrange basis function,

$$P''_s = P_s + \epsilon_{P_s}$$

and

$$\beta_{s,p} = \sum_{v \in N(s)} (D_v + \epsilon_{D_v,f}) a_{i,v}.$$

As the value of D_v and f are public, $\beta_{s,p}$ is a constant for all the shares.

By the Unisolvence theorem, $L(x)$, however, represents

$$L(x) = \Big(P''_s \times 10^{d+f} + \sum_{i=1}^{k-1} \beta_{s,p} x^i\Big) \bmod q.$$

Thus, by first putting $x = 0$ in $L(x)$, and then dividing $L(0)$ by 10^{d+f}, we recover the secret interpolated scalar as P''_s. This recovered scalar is then used as an index to the look-up tables to find the classified colors and opacities.

Note that the recovered scalar involves round-off error ϵ_{P_s}. Thus, when P''_s is used as an index to the look-up table, there can be an error $\epsilon_{C^{\uparrow}_s}$ in the classified color value $C^{\uparrow}_s$ and ϵ_{A_s} in the classified opacity A_s. In other words, using P''_s and recovered interpolated gradient G''_s, we can find the

classified color and opacity as

$$P''_s \longrightarrow C_s^{\uparrow,\prime}$$

and

$$(P''_s, G''_s) \longrightarrow A'_s,$$

where

$$C_s^{\uparrow,\prime} = C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}} \tag{6.17}$$

and

$$A'_s = A_s + \epsilon_{A_s}. \tag{6.18}$$

As the look-up tables are implemented as piecewise linear functions, even for a very small value of ϵ_{P_s} (for example, ϵ_{P_s} approaches zero), the value of $\epsilon_{C_s^{\uparrow}}$ can be as large as 255 (same argument for ϵ_{A_s}). In practice, the possibility of such high error, however, is less as ϵ_{P_s} is often unable to change the value of truncated P''_s (truncation is performed before the look-up operation), and in a look-up table, there are few spiked transitions for neighbouring entries (due to spatial coherence in an image). However, to obtain zero round-off error, we can set the value of rounding bits d and f as large as the machine precision.

Shading: Using the classified colors and the Phong illumination factors received from Interpolator, each datacenter of Compositor performs color shading. The p^{th} datacenter of Compositor, however, receives only the p^{th} share of the Phong illumination factors: $Y'_{s,p}$ and $Z'_{s,p}$. Therefore, even though the classified colors and classified opacities are known, the p^{th} datacenter can only know the p^{th} share of the shaded color.

Using Equation 6.2, the p^{th} share of the scaled shaded color can be found by

$$\begin{aligned} C'_{s,p} &= C_s^{\uparrow,\prime} Y'_{s,p} + Z'_{s,p} \\ &= (C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}}) Y'_{s,p} + Z'_{s,p} \qquad \text{(by Equation 6.17)} \end{aligned}$$

$$
\begin{aligned}
&= (C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}})\Big(\big((Y_s + \epsilon_{Y_s}) \times 10^{d+f} + \alpha_{s,p}\big) \bmod q\Big) \\
&+ \big((Z_s + \epsilon_{Z_s}) \times 10^{d+f} + \alpha_{s,p}\big) \bmod q \qquad \text{(by Equation 6.11)} \\
&\equiv \Big(\big((C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}})(Y_s + \epsilon_{Y_s}) + (Z_s + \epsilon_{Z_s})\big) \times 10^{d+f} + (C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}})\alpha_{s,p} + \alpha_{s,p}\Big) \bmod q \\
&\equiv \Big(\big((C_s^{\uparrow} Y_s + Z_s) + (C_s^{\uparrow}\epsilon_{Y_s} + \epsilon_{C_s^{\uparrow}} Y_s + \epsilon_{C_s^{\uparrow}}\epsilon_{Y_s} + \epsilon_{Z_s})\big) \times 10^{d+f} \\
&+ (C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}})\alpha_{s,p} + \alpha_{s,p}\Big) \bmod q \\
&\equiv \big((C_s + \epsilon_{C_s}) \times 10^{d+f} + (C_s^{\uparrow} + \epsilon_{C_s^{\uparrow}} + 1)\alpha_{s,p}\big) \bmod q, \qquad \text{(by Equation 6.2)}
\end{aligned} \tag{6.19}
$$

where C_s is the color shaded by conventional rendering, and

$$
\epsilon_{C_s} = C_s^{\uparrow}\epsilon_{Y_s} + \epsilon_{C_s^{\uparrow}} Y_s + \epsilon_{C_s^{\uparrow}}\epsilon_{Y_s} + \epsilon_{Z_s} \tag{6.20}
$$

is the error in shading.

Composition: In this step, each datacenter accumulates the shaded colors and classified opacities of c sample points which hold $PID_{(S,T)}$, and find the opacity and color share along the ray originated from (S, T).

The classified opacity A'_s of a sample point s is close to the opacity of s obtained in conventional rendering. Therefore, putting A'_s in Equation 6.5, we obtain O'_i as

$$
\begin{aligned}
O'_i &= A'_{s_i} \prod_{j=i+1}^{c} \big(1 - A_{s'_j}\big) \\
&= (A_{s_i} + \epsilon_{A_{s_i}}) \prod_{j=i+1}^{c} \big(1 - (A_{s_j} + \epsilon_{A_{s_j}})\big) \qquad \text{(by Equation 6.18)} \\
&= A_{s_i} \prod_{j=i+1}^{c} (1 - A_{s_j}) + A_{s_i} \prod_{j=i+1}^{c} \epsilon_{A_{s_j}} + \epsilon_{A_{s_i}} \prod_{j=i+1}^{c} \big(1 - (A_{s_j} + \epsilon_{A_{s_j}})\big) \\
&= O_i + \epsilon_{O_i}, \qquad \text{(by Equation 6.5)}
\end{aligned} \tag{6.21}
$$

where

$$
\epsilon_{O_i} = A_{s_i} \prod_{j=i+1}^{c} \epsilon_{A_{s_j}} + \epsilon_{A_{s_i}} \prod_{j=i+1}^{c} \big(1 - (A_{s_j} + \epsilon_{A_{s_j}})\big) \tag{6.22}
$$

is the error in O_i. Next, using O'_i in Equation 6.4, we obtain the composite opacity A' as

$$\begin{aligned} A' &= \sum_{i=1}^{c} O'_i \\ &= \sum_{i=1}^{c} (O_i + \epsilon_{O_i}) && \text{(by Equation 6.21)} \\ &= \sum_{i=1}^{c} O_i + \sum_{i=1}^{c} \epsilon_{O_i} \\ &= A + \epsilon_A, && \text{(by Equation 6.4)} \end{aligned} \tag{6.23}$$

where

$$\epsilon_A = \sum_{i=1}^{c} \epsilon_{O_i} \tag{6.24}$$

is the error in opacity composition.

On the other hand, the shaded color $C'_{s,p}$ of s is a share of the shaded color of s obtained in conventional rendering. Thus, the composition of these shaded color shares will also produce a share of the color composited in conventional rendering.

We, however, know that conventional color composition (given in Equation 6.3) cannot composite color shares as it involves floating point operations. In our case, the floating point operand is O'_i. Thus, we replace O'_i with a fixed point number

$$O_i^{(g),\prime} = (O'_i + \epsilon_{O'_i,g}) \times 10^g, \tag{6.25}$$

where $\epsilon_{O'_i,g}$ is the round-off error obtained in rounding off O'_i by g decimal places. Using $O_i^{(g),\prime}$ in place of O_i, and shaded color share $C'_{s,p}$ in place C_s in Equation 6.3, we obtain the p^{th} share of the scaled composite color by

$$\begin{aligned} C'_p &= \sum_{i=1}^{c} C'_{s_i,p} O_i^{(g),\prime} \\ &\equiv \sum_{i=1}^{c} \left((C_{s_i} + \epsilon_{s_i}) \times 10^{d+f} + (C^{\uparrow}_{s_i} + \epsilon_{C^{\uparrow}_{s_i}} + 1)\alpha_{s_i,p} \right) \bmod q O_i^{(g),\prime} && \text{(by Equation 6.19)} \end{aligned}$$

$$
\begin{aligned}
&\equiv \sum_{i=1}^{c} \left((C_{s_i} + \epsilon_{s_i}) O_i^{(g),\prime} \times 10^{d+f} + (C_{s_i}^{\uparrow} + \epsilon_{C_{s_i}^{\uparrow}} + 1)\alpha_{s_i,p} O_i^{(g),\prime}\right) \bmod q \\
&\equiv \Big(\sum_{i=1}^{c} (C_{s_i} + \epsilon_{s_i}) O_i^{(g),\prime} \times 10^{d+f} + \sum_{i=1}^{c} (C_{s_i}^{\uparrow} + \epsilon_{C_{s_i}^{\uparrow}} + 1)\alpha_{s_i,p} O_i^{(g),\prime} \Big) \bmod q \\
&\equiv \Big(\sum_{i=1}^{c} (C_{s_i} + \epsilon_{s_i}) O_i^{(g),\prime} \times 10^{d+f} + \Gamma_p \Big) \bmod q \\
&\equiv \Big(\sum_{i=1}^{c} (C_{s_i} + \epsilon_{s_i}) (O_i' + \epsilon_{O_i',g}) \times 10^{g} \times 10^{d+f} + \Gamma_p \Big) \bmod q \qquad \text{(by Equation 6.25)} \\
&\equiv \Big(\sum_{i=1}^{c} (C_{s_i} + \epsilon_{s_i}) (O_i + \epsilon_{O_i} + \epsilon_{O_i',g}) \times 10^{d+f+g} + \Gamma_p \Big) \bmod q \qquad \text{(by Equation 6.21)} \\
&\equiv \Big(\sum_{i=1}^{c} \left(C_{s_i} O_i + \epsilon_{s_i} O_i + (C_{s_i} + \epsilon_{s_i})(\epsilon_{O_i} + \epsilon_{O_i',g}) \right) \times 10^{d+f+g} + \Gamma_p \Big) \bmod q \\
&\equiv \Big(\sum_{i=1}^{c} C_{s_i} O_i + \sum_{i=1}^{c} \left(\epsilon_{s_i} O_i + (C_{s_i} + \epsilon_{s_i})(\epsilon_{O_i} + \epsilon_{O_i',g}) \right) \times 10^{d+f+g} + \Gamma_p \Big) \bmod q \\
&\equiv \left((C + \epsilon_C) \times 10^{d+f+g} + \Gamma_p \right) \bmod q, \qquad \text{(by Equation 6.3)} \qquad (6.26)
\end{aligned}
$$

where C is the color composited by conventional ray-casting,

$$
\epsilon_C = \sum_{i=1}^{c} \left(\epsilon_{s_i} O_i + (C_{s_i} + \epsilon_{s_i})(\epsilon_{O_i} + \epsilon_{O_i',g}) \right) \tag{6.27}
$$

is the error in composition, and

$$
\begin{aligned}
\Gamma_p &= \sum_{i=1}^{c} (C_{s_i}^{\uparrow} + \epsilon_{C_{s_i}^{\uparrow}} + 1) O_i^{(g),\prime} \alpha_{s_i,p} \\
&= \sum_{i=1}^{c} (C_{s_i}^{\uparrow} + \epsilon_{C_{s_i}^{\uparrow}} + 1) O_i^{(g),\prime} \sum_{v \in N(s_i)} (D_v + \epsilon_{D_v,f}) \alpha_{v,p} \times 10^{f} \qquad \text{(by Equation 6.12)} \\
&= \sum_{i=1}^{c} (C_{s_i}^{\uparrow} + \epsilon_{C_{s_i}^{\uparrow}} + 1) O_i^{(g),\prime} \sum_{v \in N(s_i)} (D_v + \epsilon_{D_v,f}) \sum_{j=1}^{k-1} a_{j,v} p^j \times 10^{f} \qquad \text{(by Equation 6.9)} \\
&= \sum_{j=1}^{k-1} \sum_{i=1}^{c} \sum_{v \in N(s_i)} (C_{s_i}^{\uparrow} + \epsilon_{C_{s_i}^{\uparrow}} + 1) O_i^{(g),\prime} (D_v + \epsilon_{D_v,f}) a_{j,v} \times 10^{f} \times p^j \\
&= \sum_{j=1}^{k-1} b_j p^j, \qquad (6.28)
\end{aligned}
$$

where

$$b_j = \sum_{i=1}^{c} \sum_{v \in N(s_i)} (C^{\uparrow}_{s_i} + \epsilon_{C^{\uparrow}_{s_i}} + 1) O_i^{(g),\prime} (D_v + \epsilon_{D_v,f}) a_{j,v} \times 10^f.$$

Note that b_j is a constant for all the shares as $a_{j,v}$, D_v, f, g, $C^{\uparrow}_{s_i}$, O_i, and c are public.

Since the theoretic value of classified errors $\epsilon_{C^{\uparrow}_s}$ and ϵ_{A_s} can be as large as 255 (as discussed earlier), the error in color composition ϵ_C and the error in opacity composition ϵ_A can be as large as 255. However, we have argued before that, in practice, such a case is unlikely.

Furthermore, if we set the value of d and f as large as machine precision, then we can obtain $\epsilon_{C^{\uparrow}_s} = 0$ and $\epsilon_{A_s} = 0$. By putting $\epsilon_{A_s} = 0$ in Equation 6.22, we obtain $\epsilon_{O_i} = 0$. Putting this value of ϵ_{O_i} in Equation 6.24 and Equation 6.20, we obtain $\epsilon_A = 0$ and $\epsilon_s = 0$, respectively. Finally, putting $\epsilon_{O_i} = 0$ and $\epsilon_s = 0$ in Equation 6.27, we obtain

$$\epsilon_C = \sum_{i=1}^{c} \epsilon_{O'_i,g}.$$

We know that the round-off error $\epsilon_{O'_i,g}$ satisfies $|\epsilon_{O'_i,g}| \leq 0.5 \times 10^{-f}$. Thus, for $g > 3 + t$, where t is an integer satisfying $c \leq 10^t$, we can bound ϵ_C by ± 1.

Finally, collecting the composited color C'_p, composited opacity A', and the coordinates (S_p, T_p) of all $PID_{(S,T)}$, the p^{th} datacenter constructs the p^{th} share image, and transmits this share image to the user.

Image recovery

In this step, an authorized user recovers the secret image from k share images received from any k datacenters. The opacity of a pixel in the share image is not hidden, therefore the opacity A' of the p^{th} pixel of a share image becomes the opacity of the p^{th} pixel of the secret image. The (x, y)-coordinates and colors of a pixel of the secret image, however, are obtained by Lagrange interpolation on k shares of (x, y)-coordinates (i.e., k (S_p, T_p)'s) and k color shares (i.e., k C'_p's) respectively.

From k S_p's (the same as for T_p's), the Lagrange interpolated polynomial can be found by

$$\begin{aligned} L(x) &= \sum_{i=0}^{k-1} S_{x_i} t_i(x) \bmod q' \\ &= \sum_{i=0}^{k-1} \big(S + \sum_{j=1}^{k-1} a_j x_i^j\big) t_i(x) \bmod q', \qquad \text{(by Equation 6.13)} \end{aligned}$$

where x_i is the i^{th} share number. By the Unisolvence theorem, $L(x) = (S + \sum_{i=1}^{k-1} a_i x^i) \bmod q'$. Thus, we find the secret S by setting $x = 0$ in L(x).

Similarly, the Lagrange interpolated polynomial from k color shares can be found by

$$\begin{aligned} M(x) &= \sum_{i=0}^{k-1} C'_{x_i} t_i(x) \bmod q \\ &= \sum_{i=0}^{k-1} \big((C + \epsilon_C) \times 10^{d+f+g} + \Gamma_{x_i}\big) t_i(x) \bmod q \qquad \text{(by Equation 6.26)} \\ &= \sum_{i=0}^{k-1} \big((C + \epsilon_C) \times 10^{d+f+g} + \sum_{j=1}^{k-1} b_j x_i^j\big) t_i(x) \bmod q. \qquad \text{(by Equation 6.28)} \end{aligned}$$

By the Unisolvence theorem,

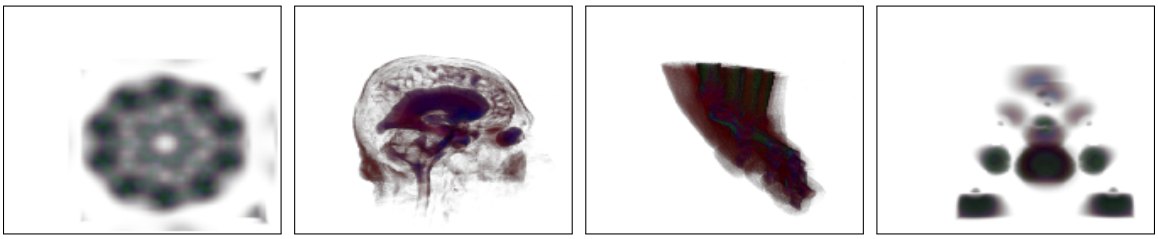
$$M(x) = \big((C + \epsilon_C) \times 10^{d+f+g} + \sum_{j=1}^{k-1} b_j x^j\big) \bmod q.$$

Thus, by first setting $x = 0$ in $M(x)$ and then dividing $M(0)$ by 10^{d+f+g}, we recover the secret color as $C + \epsilon_C$.

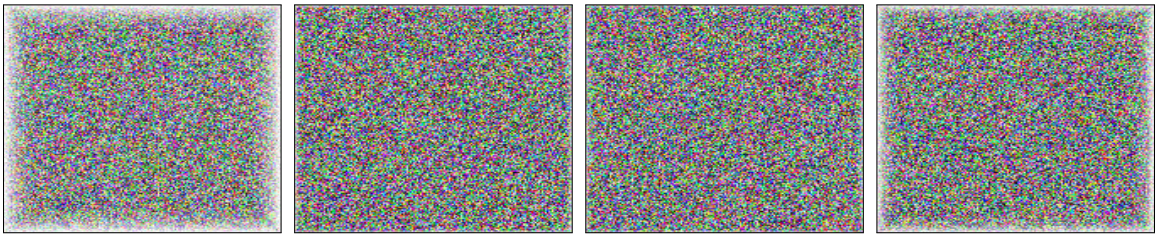
Note that as $M(0) \leq (C + \epsilon_C^{max}) \times 10^{d+f+g}$, where ϵ_C^{max} is the maximum value of ϵ_C, the prime number q must be greater than $(C + \epsilon_C^{max}) \times 10^{d+f+g}$. For example, for d and f as large as machine precision, and $g \geq 3 + t$, q must be greater than $256 \times 10^{d+f+g}$.

6.3 Results and Analysis

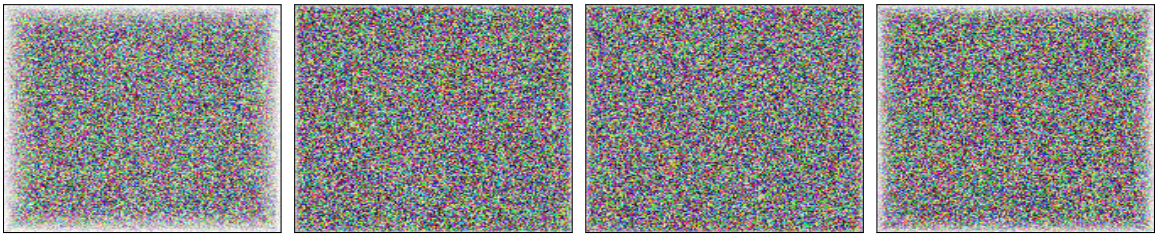
We simulated the server, the datacenters, and the client in a PC powered by an Intel Core 2 Quad 2.83 Ghz processor and with 4GB of RAM. We implemented our secure post-classification volume



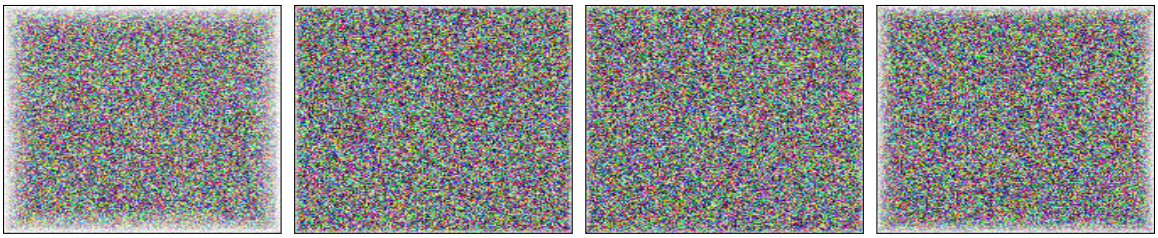
(a) Secret *Bucky* (b) Secret *Head* (c) Secret *Foot* (d) Secret *Ironprot*



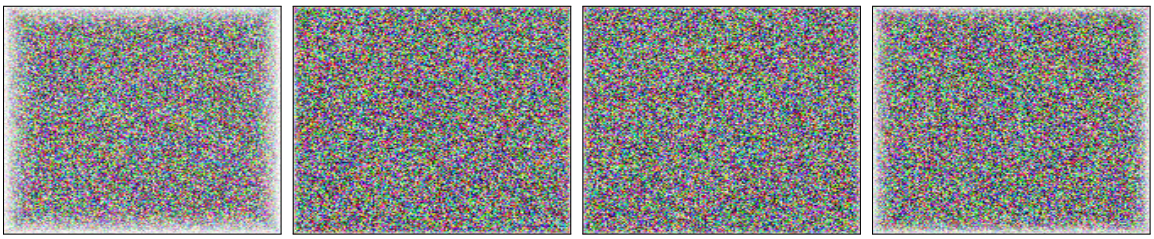
(e) First Share of *Bucky* (f) Second Share of *Head* (g) Third Share of *Foot* (h) First Share of *Ironprot*



(i) Third Share of *Bucky* (j) Third Share of *Head* (k) Fourth Share of *Foot* (l) Third Share of *Ironprot*

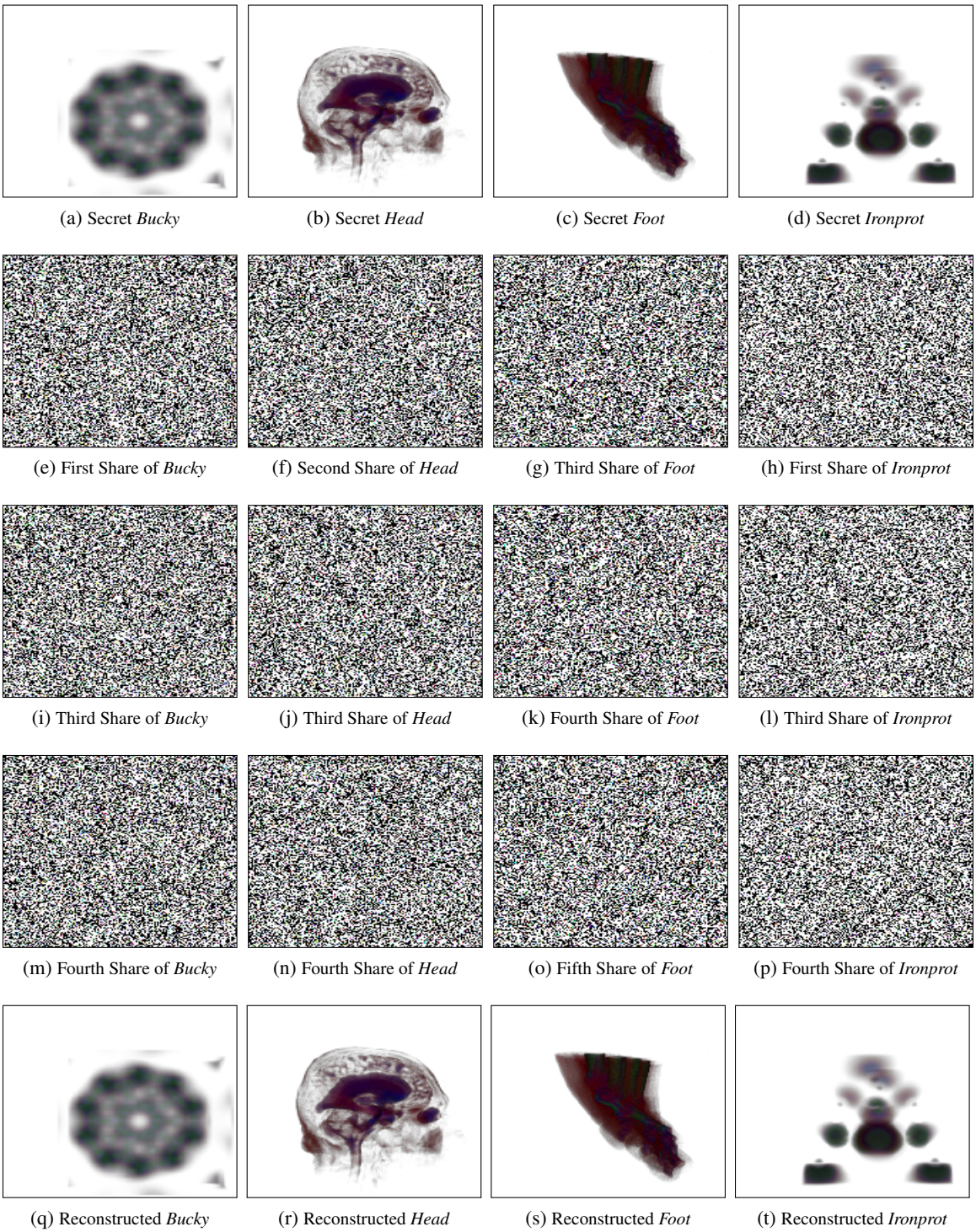


(m) Fourth Share of *Bucky* (n) Fourth Share of *Head* (o) Fifth Share of *Foot* (p) Fourth Share of *Ironprot*



(q) Reconstructed *Bucky* (r) Reconstructed *Head* (s) Reconstructed *Foot* (t) Reconstructed *Ironprot*

Figure 6.3: Rendered image in Interpolator



(a) Secret *Bucky* (b) Secret *Head* (c) Secret *Foot* (d) Secret *Ironprot*

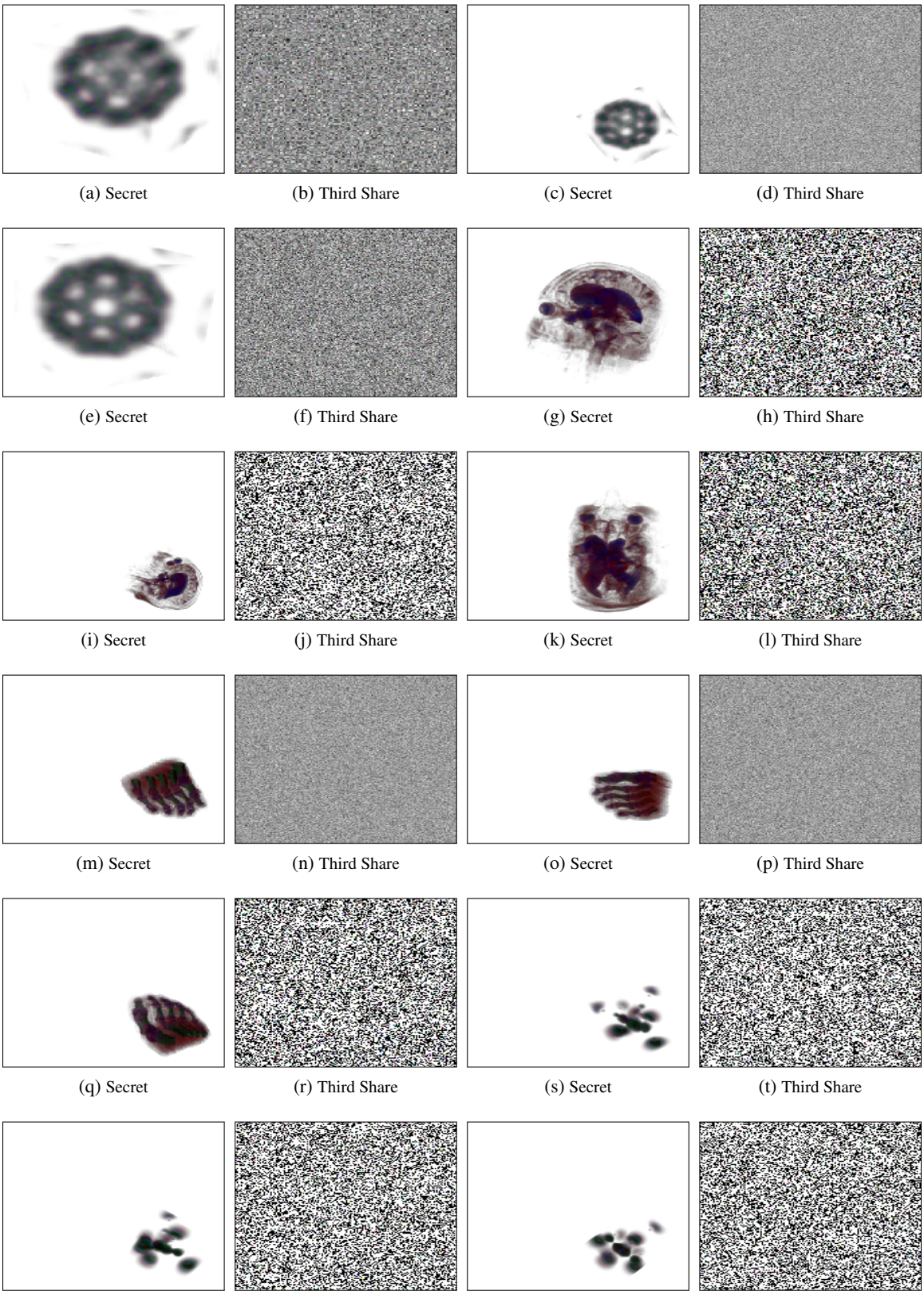
(e) First Share of *Bucky* (f) Second Share of *Head* (g) Third Share of *Foot* (h) First Share of *Ironprot*

(i) Third Share of *Bucky* (j) Third Share of *Head* (k) Fourth Share of *Foot* (l) Third Share of *Ironprot*

(m) Fourth Share of *Bucky* (n) Fourth Share of *Head* (o) Fifth Share of *Foot* (p) Fourth Share of *Ironprot*

(q) Reconstructed *Bucky* (r) Reconstructed *Head* (s) Reconstructed *Foot* (t) Reconstructed *Ironprot*

Figure 6.4: Rendered image in Compositor



(a) Secret (b) Third Share (c) Secret (d) Third Share

(e) Secret (f) Third Share (g) Secret (h) Third Share

(i) Secret (j) Third Share (k) Secret (l) Third Share

(m) Secret (n) Third Share (o) Secret (p) Third Share

(q) Secret (r) Third Share (s) Secret (t) Third Share

(u) Secret (v) Third Share (w) Secret (x) Third Share

Figure 6.5: Rendered image in Compositor from multiple view points

ray-casting by integrating $(3, 5)$ Shamir's secret sharing to the volume ray-casting module of open source 3D visualization software VTK. As our implementation of VTK allows a maximum of 64 bits to represent an integer, to convert a real number to an integer while keeping the error less, we fixed $d = 5, f = 5$, and $g = 6$. To validate our scheme, we used four sets of test volume data: *Foot*, *Head*, *Bucky*, and *Ironprot*, whose details are given in Table 5.1.

Figure 6.3 and Figure 6.4 show the secret image, share images, and the recovered image in the Interpolator and Compositor respectively. Figure 6.5 shows the results from multiple view points. As can be verified from these figures, the image available to an individual datacenter of Interpolator or Compositor is noise-like.

Note that since the color table look-up operations are performed by the Compositor, the proposed scheme does not need interaction with the server to provide user interactions requiring the modification of color look-up table.

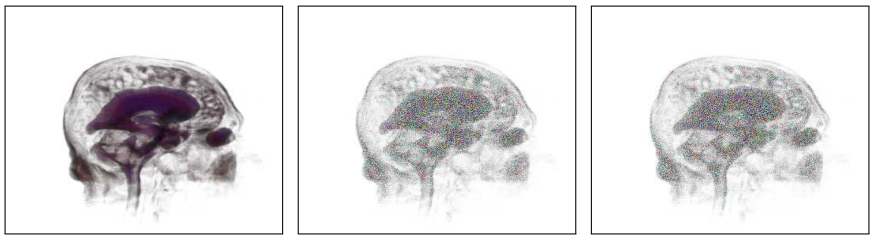
6.3.1 Security analyses

We now analyze the level of confidentiality, integrity, availability, and privacy that our scheme provides.

Confidentiality

In addition to the perceptual security, which can be verified from Figure 6.3, Figure 6.4, and Figure 6.5, our scheme, by using Shamir's secret sharing to hide the shading factors both in Interpolator and in Compositor, offers perfect secrecy for shaded colors. Furthermore, as we use Shamir's secret sharing to hide unshaded color in Interpolator and pixel positions in Compositor, a datacenter in Interpolator or in Compositor, irrespective of knowledge of pixel positions and of unshaded colors, respectively, can get little information (which is constant in a group of at most $k - 1$ datacenters) about the image. This information only helps an adversary in guessing the secret image. Without all required information, the reconstructed image, as shown Figure 6.3 for Interpolator, is also noise-like.

However, as a datacenter in Interpolator can know the secret (x, y)-coordinate of a pixel and a datacenter in Compositor can know the secret opacity and unshaded color, an adversary, by access-



(a) (b) (c)

Figure 6.6: (a), (b), and (c) are the recovered images when all three share images are tampered, two share images are tampered, and one share image is tampered respectively.

ing only one datacenter in Interpolator and one datacenter in Compositor, can know the rendered image.

Integrity

By inheriting the property of (k, n) secret sharing, our scheme ensures data/image integrity when an adversary can access at most $n - 1$ datacenters in Interpolator or Compositor.

The $k < n$ condition provides $\binom{n}{k}$ different ways to reconstruct the secret image. Therefore, if an adversary tampers with the share volume of at most $n - 1$ datacenters in Interpolator or the share images of at most $n - 1$ datacenters in Compositor, then at most $\binom{n-1}{k}$ images, which are recovered from k share images that were tampered by obeying the homomorphic property of secret sharing, can be alike (if tampering does not obey the homomorphic property, then the recovered images can be different). The remaining recovered images, which use the correct share image in image recovery, however, will be different from the images recovered from tampered images (as shown in Figure 6.6). Thus, by comparing at most $\binom{n-1}{k} + 1$ recovered images, the client can detect tampering. However, if the adversary can tamper the information of all n datacenters in Interpolator/Compositor, then our scheme fails to detect tampering. Similarly, in the case of (k, k) secret sharing, which produces only one recovered image, our scheme cannot detect tampering even when only one share image is tampered.

Availability

Due to (k, n) secret sharing, our scheme also ensures data availability as the client is able to reconstruct the secret image even if at most $n-k$ datacenters in Compositor are unable to participate. The non-participation of a datacenter in Compositor can either be due to the unavailability of necessary information from the respective datacenter in Interpolator (when a datacenter in Interpolator does not participate) or due to local issues.

6.3.2 Privacy analysis

The privacy loss is dependent on two factors: loss of identity and loss of sensitive information (e.g., the information about the diseases one is suffering from) associated with the identity. The loss of one factor in the absence of another, however, is harmless. Thus, by not disclosing the sensitive information contained in the data/image to an unauthorized user, our scheme preserves the privacy of the owner of the data/image.

6.3.3 Performance analysis

The usability of our framework is dependent on its computational overhead and data overhead, which are analyzed in this section. These costs affect the visualization latency of interacting with the rendered image. We are, however, less concerned with the overhead associated with the pre ray-projection operation since it can be performed offline.

Data overhead

Our framework introduces two types of data overhead: one during the communication of information from Interpolator to Compositor and another during the communication of k share images from Compositor to the client.

In our framework, the m^{th} datacenter in Compositor requires at least k shares of the scalar value, k shares of the gradient, and one share of the shading factors per sample point. Therefore, if b_1 bits are required to represent a share (which is equal to the number of bits required to represent the prime number q), a total of $2kb_1 + 2b_1$ number of bits are required by a datacenter in Compositor

to complete rendering. Additionally, the datacenter also must download b_2 bits to receive a share of the (x, y) coordinate of a pixel. Thus, in our implementation that uses $(3, 5)$ secret sharing, $b_1 = 64$, and $b_2 = 18$, a datacenter in Compositor must download approximately five times more data than conventional cloud-based rendering.

Similarly, the user requires k share images to recover the secret image. Thus, a total of $(3b_1 + b_2)k + 8$ bits is required to reconstruct the color and opacity of a pixel: due to which, the data overhead is $\frac{(3b_1+b_2)k-24}{32}$ times more than conventional server-side volume ray-casting. Therefore, for our implementation, the user must download approximately 19 times more data than conventional server-side volume ray-casting – such high data overhead is the main weakness of our scheme.

Computation overhead

Our scheme has two types of computation overhead: the computational overhead during rendering and the computational overhead during image recovery.

We assume that a datacenter in our scheme can render its share image as fast as a server in the conventional ray-casting can render the secret image. Therefore, the computational overhead in rendering mainly results from a datacenter in Interpolator performing $k(k-1)$ extra integer multiplications, $2(k-1)$ extra integer additions, and two integer modular prime operations to share the (x, y)-coordinate of a pixel, and creating the $PID_{(x,y)}$ for each pixel. Thus, in our simulation, which is implemented in C, a datacenter must work for an extra 94 ms for a 512×512 image space.

The computational overhead in recovering the secret image is dependent on the computation cost of the Lagrange interpolation, which is used to reconstruct the secret color value and the secret (x, y)-coordinate of a pixel, and the dimensions of the image space. Our C implementation to recover a secret image from the first, second, and third share images finds the computation cost to be approximately 172 ms for a 512×512 dimension image.

6.4 Chapter Summary

In this chapter, first, we modified post-classification volume ray-casting to perform integer-only operations by converting all floating point numbers to fixed point numbers. Then, we proposed our

secure post-classification volume ray-casting framework that hides both the color and opacity information from a 3D volumetric data from a datacenter by using Shamir's secret sharing, allows a datacenter to render a noise-like share image from a share volume, and allows a user to recover the secret image from the share images. Although this framework provides full perceptual confidentiality by hiding both color and shape of an object from a datacenter, high data overhead to a user, however, is a concern.

Chapter 7

Conclusion and Future Work

Cloud-based imaging presents security and privacy concerns. Although these concerns have been addressed for cloud-based volume data/image storage, they are still an issue for cloud-based volume data/image processing, such as image scaling/cropping or volume ray-casting. Securing cloud-based volume data/image processing is challenging since: (i) no cryptosystem can hide all operations of a volume data/image processing algorithm with acceptable overhead, and (ii) the floating point operations of a volume data/image processing algorithm are incompatible with the modular prime operations of a cryptosystem.

This thesis addressed the security and privacy issues in three popular cloud-based image processing schemes: cloud-based image scaling/cropping, cloud-based pre-classification volume ray-casting, and cloud-based post-classification volume ray-casting, by using Shamir's secret sharing (and its variants) to hide the additions and scalar multiplications operations of these algorithms. Other operations were either pre-computed or distributed among datacenters such that none of the datacenters gets enough information to know the secret volume data/image. To solve the incompatibility issue of Shamir's secret sharing with a volume data/image processing algorithm, we either excluded the modular prime operation from secret sharing or converted floating point operations to fixed point operations. We showed that the former technique degrades security, and the latter loses some information from the rendered image.

Our secure image scaling/cropping framework (Chapter 4) uses a new $(3, k, n)$ ramp secret image sharing scheme to share an image. This image sharing scheme allows scaling/croping of noise-

like shadow images stored in third-party datacenters. Receiving at least k scaled/cropped shadow images, a user can recover the secret scaled/cropped image. Experiments and analyses showed that this framework can hide the content of an image in a datacenter, detect tampering of an image performed at a datacenter, and withstand the breakdown of certain number of datacenters. This framework results in low computation overhead, but significant data overhead (1.5 times more data overhead than conventional image streaming) to the user.

Our secure cloud-based pre-classification volume ray-casting framework (Chapter 5) hides the color information of volume data from datacenters. This color-hidden data renders to color-hidden images, which can be used by a user to recover the secret rendered image. Since Shamir's secret sharing is non-homomorphic to opacity multiplications, this scheme, however, cannot hide the shape of an object. To address the incompatibility of secret sharing with ray-casting, we modified either secret sharing (which is the case in SR-MSSS) or ray-casting (which is the case in SR-MSSS), and to decrease the high data overhead of both these techniques, we optimized modified secret sharing (which is the case in SR-RSS). Experiments and analyses showed that among our three frameworks, SR-MPVR provides high security at the cost of high data overhead and the loss of some information from the rendered image; SR-MSSS renders a lossless image and provides moderate security at the cost of moderate data overhead; and SR-RSS incurs low data overhead at the cost of high security and some information loss from the rendered image.

Finally, we proposed secure post-classification volume ray-casting (Chapter 6), which hides both the color and shape of an object from rendering datacenters. This scheme distributes the rendering tasks among a number of datacenters and integrates Shamirs secret sharing into the rendering pipeline in such a way that none of the datacenters can know the secret data or the rendered image. A user, however, can recover the secret image from the hidden images. Experiments and analyses showed that this framework can also detect tampering of data/image performed at a datacenter, and can withstand breakdown of certain number of datacenters. This framework results in low computation cost and high data overhead (19 times more data overhead than conventional rendering) to the user.

To the best of our knowledge, we are the first to apply Shamir's secret sharing based secure multiparty computation to 2D image scaling/cropping and volume ray-casting algorithms. In this

work, we mainly studied the feasibility of applying secret sharing (or its variant) to an imaging algorithm, and designed practical secure frameworks. At this point, we were less concerned about performance and image quality.

Future works, however, may focus on improving the performance and image quality of the proposed frameworks. Furthermore, the presented ideas can be applied to secure other areas such as secure video scaling/cropping, secure surface rendering etc. In the following, we briefly discuss some of the possible future works.

7.1 Improvement of the Proposed Frameworks

In this section, we discuss one improvement each for secure cloud-based image scaling/cropping, secure cloud-based pre-classification volume ray-casting, and secure cloud-based post-classification volume ray-casting.

7.1.1 Secure scaling/cropping of a compressed images

Our secure cloud-based image scaling/cropping framework scales and crops an uncompressed shadow image. Thus, a datacenter must either store an uncompressed image, or decompress a compressed shadow image before scaling/cropping. Evidently, The former approach results in high data overhead. The latter approach, which is typically used in conventional image scaling/cropping, is not much more improved than the former since the spatial coherence of an image is destroyed in a noise-like shadow image. As a result, the proposed framework requires significantly more storage and bandwidth than conventional image streaming.

One can decrease the storage and bandwidth requirements by secret sharing a compressed image, and allowing scaling/cropping of a shadow image of the compressed image. Since compression techniques differ from one another, a framework involving one compression technique can present different challenges than a framework involving another compression technique. For JPEG compression, we briefly discuss the challenges below.

Secret sharing a JPEG compressed image [99], and scaling/cropping a secret JPEG compressed image [100, 101] have been independently studied. Proposed secret sharing schemes and proposed

scaling/cropping schemes simultaneously hide and scale the DCT coefficients. Thus, an obvious approach can be to combine these schemes, and hide the DCT coefficients such that scaling can be performed on the hidden coefficients. As described below, such an approach, however, is not trivial.

To hide DCT coefficients, existing secret sharing schemes secret share DC coefficients, and either permute or randomize the AC coefficients [99]. Permutation of AC coefficients does not support cropping, and using the DC coefficient as a seed to randomize AC coefficients does not support scaling. Furthermore, hiding all the AC coefficients makes zig-zag coding inefficient, and therefore increases the size of the compressed image. Thus, a new secret sharing scheme must be designed.

An obvious approach can be to secret share all the AC coefficients, but use one secret sharing polynomial for all the coefficients having the same value. We believe that this scheme can produce noise-like images, and does not disturb zig-zag coding. The discloser of the number of AC coefficients and their positions, however, can weaken security since the value of some high frequency coefficients can be guessed. Thus, a detailed analysis of this approach and the possible tradeoffs need to be examined.

7.1.2 Hiding the shape of an object in secure pre-classification ray-casting

Our secure pre-classification volume ray-casting cannot hide the shape of an object from a datacenter, and therefore does not provide high data confidentiality. We know that the shape can be hidden only when the opacities are hidden, and Shamir's secret sharing is non-homomorphic to the multiplications in opacity rendering. Thus, new techniques must be devised to hide both colors and opacities. In the following, we discuss two preliminary ideas.

First, we can use the idea of a secure post-classification volume ray-casting framework to separate color rendering from opacity rendering such that the color renderer (i.e., the group of datacenters rendering the color) does not know the secret opacities, and the opacity renderer does not know the secret pixel positions. Instead, shares of opacities can be provided to the color renderer, and proxy pixel positions can be provided to the opacity renderer. Executing such an idea, however, is challenging since the direction of the projected rays cannot be hidden during opacity interpolation.

Alternatively, the opacities can be hidden by using a cryptosystem that is homomorphic to unlimited multiplications. Cryptosystems such as ElGamal encryption, which are homomorphic to a

predefined number of multiplications, can fulfill this requirement with high overheads. In future, it will be interesting to examine if such a cryptosystem can be used in combination with Shamir's secret sharing.

7.1.3 Using Phong shading in post-classification ray-casting

Our secure post classification volume ray-casting uses Gouard shading, and therefore renders inferior color than volume ray-casting using Phong shading. Thus, in the future, one can improve our scheme by supporting Phong shading.

Phong shading computes the illumination factors

$$Y_s = k_a + k_d MAX(\mathcal{N}_s.L, 0)$$

and

$$Z_s = k_s MAX\big((\mathcal{N}_s.R)^n, 0),$$

of a sample point after the projection of rays. Using these illumination factors, colors are then shaded by

$$C_s = Y_s C_s^{\uparrow} + Z_s.$$

Thus, to hide the shaded color, both Y_s and Z_s must be hidden. We can hide Y_s and Z_s by hiding at least one variable in the computation of Y_s and Z_s. The ambient coefficient k_a, diffuse coefficient k_d, specular coefficient k_s, and specular shininess n can be assumed to be public since they can be known by knowing the type of object being rendered. As a result, we have to hide either normal N_s, or light L and reflected light R to secure Phong shading. Both these options can be explored in future works.

7.2 Secure Video Scaling/Cropping

Cloud-based video storage/processing, such as cloud-based video conferencing and cloud-based video surveillance, are becoming popular. In this technique, videos, which can contain confidential information, are stored and processed at third-party cloud datacenters.

Although cloud-based video storage/processing can be advantageous than conventional server-side video storage/processing, security and privacy are the main concerns. For example, by accessing a datacenter, an adversary can know the participants and discussed confidential information of a video conference, or can know the identity of a person on surveillance. One can, however, address the security and privacy concerns by hiding the content of a video from a datacenter.

Two important operations on video are scaling and cropping. Downloading a large video, such as a surveillance video, may not anyways be feasible. Users connected through different devices may request video at different scale levels. Furthermore, users may just want to view a particular region of interest in the video, in which case, a cropped region should be downloaded. These two operations, scaling and cropping, can be combined to support zooming and panning, two natural user interactions. Thus, video scaling and cropping must be supported by a datacenter.

Video scaling/cropping [102], and video hiding [103, 104] are two independently well-studied areas. To the best of our knowledge, scaling/cropping a hidden video, however, has not been done yet. In the future, one may work on addressing this problem by extending our idea of scaling/cropping a hidden image.

7.3 Secure Surface Rendering

The cloud-based surface rendering framework, also called cloud-based indirect volume rendering, uses surface rendering to remotely render a 3D image. In this framework, an organization captures a set of 2D images and sends the captured images to a datacenter. Upon a user's request, the datacenter renders a 3D image from the image set by using a surface rendering algorithm, and sends the rendered image to the user. Although use of cloud datacenters relieves an organization from the complex rendering tasks, disclosure of confidential 2D images to datacenters creates security and privacy concerns.

Securing surface rendering is more challenging than direct volume rendering since surface rendering performs more complex operations. To render a 3D image, surface rendering first extracts isosurfaces from an image set, and then renders the isosurfaces by an isosurface rendering algorithm. The isosurfaces are typically extracted by marching cube or the marching tetrahedra algorithm, and

the extracted isosurfaces are rendered by the volume ray-casting or rasterization algorithm. Thus, to secure surface rendering, both the isosurface extraction algorithm and isosurface rendering algorithm must be secured. Furthermore, since hidden isosurfaces need to be input to the isosurface rendering algorithm, our secure volume ray-casting techniques cannot be used.

Bibliography

[1] VIDAR Systems Corporation. The transition to digital imaging in medicine. White Paper, 2010. http://www.vidar.com/film/images/stories/PDFs/newsroom/DigitalTransition~White~Paper~hi-res~GFIN.pdf.

[2] Ronald S. Weinstein, Anna R. Graham, and Lynne C. Richter et al. Overview of telepathology, virtual microscopy, and whole slide imaging: prospects for the future. *Human Pathology*, 40:1057–1069, August 2009.

[3] David Green. Using digital images in teaching and learning: perspectives from liberal arts institutions. Online Report, 2006. http://www.academiccommons.org/imagereport.

[4] Fabrizio Lamberti and Andrea Sanna. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 13:247–260, March 2007.

[5] Julien Jomier, Sebastien Jourdain, Utkarsh Ayachit, and Charles Marion. Remote visualization of large datasets with MIDAS and ParaViewWeb. In *Proceedings of the 16th International Conference on 3D Web Technology*, pages 147–150, Paris, France, 2011.

[6] Klaus J. Enge, Thomas Ertl, and Peter Hastreiter et al. Combining local and remote visualization techniques for interactive volume rendering in medical applications. In *Proceedings of the Conference on Visualization*, pages 449–452, Salt Lake City, Utah, United States, 2000.

[7] Olcay Sertel, Jun Kong, Hiroyuki Shimada, and et al. Computer-aided prognosis of neuroblastoma on whole-slide images: classification of stromal development. *Pattern Recognization*, 42:1093–1103, June 2009.

[8] Mikhail Smelyanskiy, Daid Holmes, and Jatin Chhugani et al. Mapping high-fidelity volume rendering for medical imaging to CPU, GPU and many-core architectures. *IEEE Transactions on Visualization and Computer Graphics*, 15:1563–1570, November 2009.

[9] Mikael Lundin, Janusz Szymas, and Ewert Linder et al. A European network for virtual microscopydesign, implementation and evaluation of performance. *European Society of Pathology*, 454:421–429, April 2009.

[10] 3DI. Cloud based medical image management and visualization platform. Online Report, 2012. `http://www.shina-sys.com/assets/brochures/3Di.pdf`.

[11] Karlheinz Dorn, Vladyslav Ukis, and Thomas Friese. A cloud-deployed 3D medical imaging system with dynamically optimized scalability and cloud costs. In *Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 155–158, Oulu, Finland, 2011.

[12] Denis Vazhenin. Cloud-based Web-service for Health 2.0. In *Proceedings on Joint International Conference on Human-Centered Computer Envirnment*, pages 240–243, Hamamatsu, Japan, 2012.

[13] Medical imaging in the cloud. Online Report, 2012. `http://www.corp.att.com/healthcare/docs/medical_imaging_cloud.pdf`.

[14] Lawrence M Kaufman. Data security in the world of cloud computing. *IEEE Security and Privacy*, 7:61–64, July 2009.

[15] Seny Kamara and Kristin Lauter. Cryptographic cloud storage. In *Proceedings of the 14th International Conference of Financial Cryptography and Data Security: Workshop on Real-Life Cryptographic Protocols and Standardization*, pages 136–149, Canary Islands, Spain, 2010.

[16] Shucheng Yu, Cong Wang, Kui Ren, and Worcester Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th Conference on Information Communications*, pages 1–9, San Diego, California, USA, 2010.

[17] Mohammed A. AlZain, Eric Pardede, Ben Soh, and James A. Thom. Cloud computing security: from single to multi-clouds. In *Proceedings of the 45th Hawaii International Conference on System Sciences*, pages 5490–5499, Hawaii, USA, 2012.

[18] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, pages 113–124, Chicago, USA, 2011.

[19] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012. `http://eprint.iacr.org/2012/144.pdf`.

[20] Kevin Henry. The theory and applications of homomorphic cryptography. Master Thesis, 2008.

[21] Erkay Savaş and Çetin Kaya Koç. Finite field arithmetic for cryptography. *IEEE Circuits and Systems*, 10:40–56, May 2010.

[22] Josh C. Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proceedings of the Advances in Cryptology*, pages 251–260, Santa Barbara, USA, 1987.

[23] George R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Arlington, USA, 1979.

[24] Lein Harn and Lin Changlu. Authenticated group key transfer protocol based on secret sharing. *IEEE Transactions on Computers*, 59:842–846, 2010.

[25] Joan Cooper, Diane Donovan, and Jennifer Seberry. Secret sharing schemes arising from Latin sqaures. *Bulletin of the Institute of Combinatorics and Its Applications*, 12:33–43, 1994.

[26] Timothy Finamore. Shamir's secret sharing scheme using floating point arithmetic. Master Thesis – Florida Atlantic University, 2012.

[27] Manoranjan Mohanty, Pradeep K. Atrey, and Wei Tsang Ooi. Secure cloud-based medical data visualization. In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 1105–1108, Nara, Japan, 2012.

[28] Octavian Catrina and Amitabh Saxena. Secure computation with fixed-point numbers. In *Proceedings of the 14th International Conference on Financial Cryptography and Data Security*, pages 35–50, Tenerife, Spain, 2010.

[29] Manoranjan Mohanty, Wei Tsang Ooi, and Pradeep K. Atrey. Scale me, crop me, know me not: supporting scaling and cropping in secret image sharing. In *Proceedings of the 2013 IEEE International Conference on Multimedia & Expo*, San Jose, USA, 2013.

[30] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.

[31] Manoranjan Mohanty, Wei Tsang Ooi, and Pradeep K. Atrey. Secure cloud-based volume data visualization. *Under Submmision in the IEEE Transaction on Information Forensics and Security*, 2013.

[32] Manoranjan Mohanty, Wei Tsang Ooi, and Pradeep K. Atrey. Secure cloud-based volume ray-casting. In *Proceedings of the 5th IEEE Conference on Cloud Computing Technology and Science*, Bristol, UK, 2013.

[33] Chia-Chi Teng, Jonathan Mitchell, and Christopher Walker et al. A medical image archive solution in the cloud. In *Proceedings of the 1st International Conference on Software Engineering and Service Sciences*, pages 431–434, Utah, USA, 2010.

[34] James Philbin, Fred Prior, and Paul Nagy. Will the next generation of PACS be sitting on a cloud? *Journal of Digital Imaging*, 24:179–183, April 2011.

[35] Chenghao He, Xi Jin, Zhanxiang Zhao, and Tian Xiang. A cloud computing solution for hospital information system. In *Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, pages 517–520, Xiamen, China, 2010.

[36] PACS in the cloud - revolutionising medical imaging. Online Report, 2013. `http://click.accenture.com/article/pacs-cloud-revolutionising-medical-imaging/`.

[37] eWEEK.com. Cloud computing in health care to reach $5.4 billion by 2017: report. Online Report, 2012. http://www.eweek.com/c/a/Health-Care-IT/Cloud-Computing-in-Health-Care-to-Reach-54-Billion-by-2017-Report-512295/.

[38] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, pages 103–114, Chicago, USA, 2009.

[39] Jeremie Tharaud, Sven Wohlgemuth, and Isao Echizen et al. Privacy by data provenance with digital watermarking. In *Proceedings of the 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 510–513, Darmstadt, Germany, 2010.

[40] Tim Rostrom and Chia-Chi Teng. Secure communications for PACS in a cloud environment. In *Proceedings of the 33rd Annual International Conference of the IEEE EMBS*, pages 8219–8221, Boston, USA, 2011.

[41] KDDI Inc. Medical real-time 3D imaging solution. Online Report, 2012. http://www.kddia.com/en/sites/default/files/file/KDDI_America_Newsletter_August_2012.pdf.

[42] Xinde Sun. A blind digital watermarking for color medical images based on PCA. In *Proceedings of the IEEE International Conference on Wireless Communications, Networking and Information Security*, pages 421–427, Beijing, China, August 2010.

[43] Hussain Nyeem, Wageeh Boles, and Colin Boyd. A review of medical image watermarking requirements for teleradiology. *Journal of Digital Imaging*, pages 1–18, September 2012.

[44] Baisa L. Gunjal and Suresh N. Mali. ROI based embedded watermarking of medical images for secured communication in telemedicine. *International Journal of Computer and Communication Engineering*, 6:293–298, June 2012.

[45] N. K. Pareeka, Vinod Patidar, and K. K. Sud. Image encryption using chaotic logistic map. *Image and Vision Computing*, 24:926–934, September 2006.

[46] Nuha O. Abokhdair, Azizah B. A. Manaf, and Mazdak Zamani. Integration of chaotic map and confusion technique for color medical image encryption. In *Proceedings of the 6th International Conference on Digital Content, Multimedia Technology and its Applications*, pages 20–23, Seoul, South Korea, 2010.

[47] Chih-Ching Thien and a Chen Lin. Secret image sharing. *Computers and Graphics*, 26:765–770, October 2002.

[48] Moni Naor and Adi Shamir. Visual cryptography. In *Proceeding of Eurocrypt*, pages 1–12, Berlin, Germany, 1994.

[49] Maurice Mignotte. How to share a secret. In *Proceedings of the 1982 Conference on Cryptography*, pages 371–375, Burg Feuerstein, Germany, 1983.

[50] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transaction on Information Theory*, 29:208–210, September 2006.

[51] Young-Chang Hou. Visual cryptography for color images. *Pattern Recognition*, 36:1619–1629, July 2003.

[52] Hao-Kuan Tso. Sharing secret images using Blakley's concept. *Optical Engineering*, 47(7):077001–3, 2008.

[53] İlker Nadi Bozkurt, Kamer Kaya, Ali Aydin Selçuk, and Ahmet M. Güloğlu. Threshold cryptography based on Blakley secret sharing. In *Proceedings of the Information Security and Cryptology*, pages 313–317, Ankara, Turkey, 2008.

[54] Shyong-Jian Shyu and Ying-Ru Chen. Threshold secret image sharing by Chinese remainder theorem. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, pages 1332–1337, Yilan, Taiwan, 2008.

[55] Sorin Iftene. General secret sharing based on the Chinese remainder theorem with applications in e-voting. *Electronic Notes in Theoretical Computer Science*, 186(0):67–84, 2007.

[56] Mustafa Ulutas., Vasif V. Nabiyev., and Guzin Ulutas. A new secret image sharing technique based on Asmuth Bloom's scheme. In *Proceedings of the International Conference on Application of Information and Communication Technologies*, pages 1–5, Baku, Azerbaijan, 2009.

[57] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing*, pages 101–111, Puerto Vallarta, Mexico, 1998.

[58] George R. Blakley and Catherine Meadows. Security of ramp schemes. In *Proceedings on Advances in Cryptology*, pages 242–268, Sanata Barbara, USA, 1985.

[59] Hirosuke Yamamoto. Secret sharing system using (k, L, n) threshold scheme. *Electronics and Communications in Japan, Part I*, 69:46–54, September 1986.

[60] Benny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. *Journal of Cryptology*, 6:87–96, 1989.

[61] Li. Bai. A reliable (k, n) image secret sharing scheme. *IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 0:31–36, October 2006.

[62] Li Li and Ahmed A. Abd El-Latif. A novel secret image sharing scheme based on chaotic system. In *Proceedings of SPIE 4th International Conference on Digital Image Processing*, Kuala Lumpur, Malaysia, 2012.

[63] Mustafa Ulutas, Güzin Ulutas, and Vasif V. Nabiyev. Medical image security and EPR hiding using Shamir's secret sharing scheme. *Journal of Systems and Software*, 84:341–353, March 2011.

[64] Saeed S. Alharthi and Pradeep K. Atrey. Further improvements on secret image sharing scheme. In *Proceedings of the 2nd ACM workshop on Multimedia in Forensics, Security and Intelligence*, pages 53–58, Firenze, Italy, 2010.

[65] Chih-Ching Thien. An image-sharing method with user-friendly shadow images. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:1161–1169, December 2003.

[66] Chang-Chou Lin and Wen-Hsiang Tsai. Secret image sharing with steganography and authentication. *Journal of System and Software*, 73:405–414, November 2004.

[67] Ran-Zan Wang and Shyong-Jian Shyu. Scalable secret image sharing. *Image Communication*, 22:363–373, April 2007.

[68] Wang Liu and Chunhui Zhao. Digital watermarking for volume data based on 3D-DWT and 3D-DCT. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pages 352–357, Seoul, Korea, 2009.

[69] Esam Elsheh and A. Ben Hamza. Secret sharing approaches for 3D object encryption. *Expert Systems with Applications*, 38:13906 – 13911, October 2011.

[70] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Maryland, USA, 2009.

[71] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. `http://eprint.iacr.org/`.

[72] Cong Wang, Ning Cao, Jin Li, Kui Ren, and Wenjing Lou. Secure ranked keyword search over encrypted cloud data. In *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems*, pages 253–262, Genoa, Italy, 2010.

[73] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Fuzzy keyword search over encrypted data in cloud computing. In *Proceedings of the 2010 IEEE International Conference on Computer Communications*, pages 1–5, San Diago, USA, 2010.

[74] Jingwei Li, Chunfu Jia, Jin Li, and Zheli Liu. A novel framework for outsourcing and sharing searchable encrypted data on hybrid cloud. In *Proceedings of the 4th International Conference on Intelligent Networking and Collaborative Systems*, pages 1–7, Bucharest, Romania, 2012.

[75] Cong Wang, Kui Ren, and Jia Wang. Secure and practical outsourcing of linear programming in cloud computing. In *Proceedings of the 30th IEEE International Conference on Computer Communications*, pages 820–828, Shanghai, China, 2011.

[76] Qingji Zheng and Xinwen Zhang. Multiparty cloud computation. *Computing Research Repository*, 1206.3717:1–8, June 2012.

[77] Durgesh K. Mishra. Tutorial: secure multiparty computation for cloud computing paradigm. In *Proceedings of the 2010 Second International Conference on Computational Intelligence, Modelling and Simulation*, pages xxiv–xxv, Bali, Indonesia, 2010.

[78] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *The Journal of Privacy and Confidentiality*, 1:59–98, April 2009.

[79] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. `http://eprint.iacr.org/`.

[80] Zekeriya Erkin, Martin Franz, and Jorge Guajardo et al. Privacy-preserving face recognition. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, pages 235–253, 2009.

[81] Erman Ayday, Jean L. Raisaro, Jean-Pierre Hubaux, and Jacques Rougemont. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, pages 95–106, 2013.

[82] R. (Inald) L. Lagendijk, Zekeriya Erkin, and Mauro Barni. Encrypted signal processing for privacy protection: conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, 2013.

[83] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the* 23*rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, USA, 1982.

[84] Andrew C. Yao. How to generate and exchange secrets. In *Proceedings of the* 27*th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Canada, 1986.

[85] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 148–164, Santa Barbara, USA, 1999.

[86] Lein Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *Computers and Digital Techniques*, 141(5):307–313, 1994.

[87] Matthew Roughan and Yin Zhang. Secure distributed data-mining and its application to large-scale network measurements. *Computer Communication Review*, 36:7–14, January 2006.

[88] Peter Bogetoft, Dan L. Christensen, and Ivan Damgård et al. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, pages 325–343. Springer Berlin Heidelberg, 2009.

[89] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pages 250–267, Tokyo, Japan, 2009.

[90] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *Advances in Cryptology*, pages 465–482, Santa Barbara, USA, 2010.

[91] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York, USA, 1987.

[92] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the Web: computing without simultaneous interaction. In *Proceedings of the 31st Annual Conference on Advances in Cryptology*, pages 132–150, Santa Barbara, CA, 2011.

[93] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay – a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium*, pages 20–20, San Diego, CA, 2004.

[94] Ramin Shahidi. Surface rendering versus volume rendering in medical imaging: techniques and applications. In *Proceedings of the Conference on Visualization*, pages 439–440, San Francisco, California, USA, 1996.

[95] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Application*, 8:29–37, May 1988.

[96] Craig M. Wittenbrink, Tom Malzbender, and Michael E. Goss. Opacity-weighted color interpolation for volume sampling. In *IEEE Symposium on Volume Visualization*, pages 135–142, North Carolina, USA, 1998.

[97] Mukesh Saini, Pradeep K. Atrey, Sharad Mehrotra, and Mohan Kankanhalli. Anonymous surveillance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6, Barcelona, Spain, 2011.

[98] Elaine M. Newton, Latanya Sweeney, and Bradley Malin. Preserving privacy by de-identifying face images. *IEEE Transaction on Knowledge and Data Engineering*, 17:232–243, Feburary 2005.

[99] Chang-Chou Lin and Wen-Hsiang Tsai. Secret image sharing with capability of share data reduction. *Optical Engineering*, 42:2340–2345, August 2003.

[100] Brian C. Smith and Lawrence A. Rowe. Algorithms for manipulating compressed images. *IEEE Computer Graphics and Applications*, 13:34–39, September 1993.

[101] Carlos L. Salazar and Trac D. Tran. On resizing images in the DCT domain. In *Proceedings of the IEEE International Conference on Image Processing*, pages 2797–2800, Singapore, 2004.

[102] Quang Minh Khiem Ngo, Guntur Ravindra, Axel Carlier, and Wei Tsang Ooi. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proceedings of the 1st Annual ACM SIGMM Conference on Multimedia Systems*, pages 259–270, Phoenix, Arizona, USA, 2010.

[103] Ahmet M. Eskicioglu, Scott Dexter, and Edward J. Delp III. Protection of multicast scalable video by secret sharing: simulation results. In *Proceedings of the Security and Watermarking of Multimedia Contents V*, pages 505–515, Santa Clara, USA, 2003.

[104] Changgui Shi and Bharat Bhargava. A fast MPEG video encryption algorithm. In *Proceedings of the 6th ACM International Conference on Multimedia*, pages 81–88, Bristol, United Kingdom, 1998.