

Multi-Input Functional Encryption and Obfuscation

A Thesis
Presented to
The Established Interdisciplinary Committee for Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Sage R. Michaels

May 2018

Approved for the Division
(Mathematics)

Dylan McNamee

Acknowledgements

I want to thank a few people.

Abstract

This is an example of a thesis setup to use the reed thesis document class.

Table of Contents

Introduction	1
Chapter 1: Background	3
1.1 Encryption	3
1.1.1 Classical Encryption	3
1.1.2 Functional Encryption	4
1.2 Obfuscation	7
1.2.1 Black Box Obfuscation	7
1.2.2 Indistinguishability Obfuscation	7
1.3 Bilinear Maps	8
1.3.1 Definition	8
1.4 Intuition	9
Chapter 2: Multi-Linear Maps	11
2.1 Cryptographic Multilinear Maps	11
2.2 Graded Encoding Schemes	11
2.2.1 Definition	11
2.3 Intuition	12
2.3.1 Structural Intuition	12
2.3.2 Encodings	12
2.4 Candidate Groups/Quotient Rings/Fields	13
Chapter 3: Indistinguishability Obfuscation	15
3.1 Notes	15
3.2 Construction	15
3.3 Usage, Limitations, and Goals	15
Chapter 4: Multi-Party Input Functional Encryption	17
4.1 Scheme	17
4.2 Construction	17
4.3 Limitations and Goals	17
Chapter 5: A Brief Introduction to the 5-GenC library	19
5.1 The DSL	19
5.2 Circuits and Branching Programs	19

5.3	Base and MMaps	19
Chapter 6:	Experiments	21
6.1	Comparison Circuit	21
6.2	Runtime Evaluation	21
Chapter 7:	Conclusion	23
References	25

Introduction

Chapter 1

Background

1.1 Encryption

Encryption a way to share a message so that only the intended recipient(s) of that message are able to read it. Historically this was done by means of obscurity, in the sense that correspondents assumed only they knew the specific method by which messages between them would be encrypted. The problem with encryption via obscurity is that as soon as a method of encryption becomes popular, it immediately becomes obsolete.

1.1.1 Classical Encryption

Now, cryptographers work to develop encryption schemes that are computationally infeasible for adversaries to break even if the method of encryption is known (this is known as Kerckhoff's Principle). To do this, encryption functions are made public but take an extra parameter that is kept secret, we call this secret a key. The best keys are ones that are decently long, so that every possible key can't be attempted by an adversary; and chosen randomly, so they are nearly impossible to guess.

In defining an encryption scheme we call the set of all valid keys K called a key space, a set of all valid messages M called the message space, and a set of all valid cipher texts (encrypted messages) C called a cipher space.

Definition 1 (Classical Encryption Scheme).

$$Gen : \mathbb{Z} \rightarrow K \times K$$

Defined to be for $\lambda \in \mathbb{Z}$, $Gen(\lambda) \rightarrow (pk, sk)$ where pk and sk are random keys of length λ .

$$Enc : K \times M \rightarrow C$$

Defined for key $pk \in K$ and message $m \in M$; to be $Enc_{pk}(m) \rightarrow c$ for some cipher text $c \in C$

$$Dec : K \times C \rightarrow M$$

Defined for key $sk \in K$ and cipher text $c \in C$; to be $Dec_{sk}(c) \rightarrow m$ for some message $m \in M$

If $sk = pk$ this is called a symmetric or private key encryption scheme meaning only the correspondents know the key and they keep it secret. If $sk \neq pk$ then this is called an asymmetric or public key encryption scheme where sk is a secret key and pk is a public key. In a public key encryption scheme anyone can encrypt a message since the public key is public, but only people with the secret key are able to decrypt.

Definition 2 (Correctness). In this setting we say an encryption scheme Π is **correct** if for $n \in \mathbb{Z}$, $(sk, pk) \leftarrow Gen(n)$ and $m \in M$

$$Dec_{sk}(Enc_{pk}(m)) = m$$

Suppose Alice wants to send Bob a secret message m . To do this Bob would have to run $Gen(n) \rightarrow pk, sk$ and then send pk to Alice. Then Alice gets $c := Enc_{pk}(m)$ and sends c over to Bob. Finally Bob gets $m' := Dec_{sk}(c)$. If the scheme is correct then $m' = m$ and Bob is able to read Alice's message. The above interaction is represented in the following diagram.

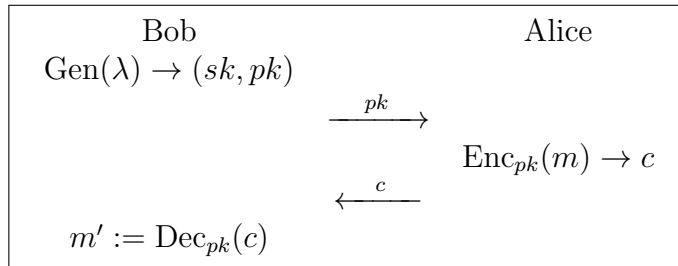


Table 1.1: Public Key Encryption Protocol

1.1.2 Functional Encryption

With classical encryption, decryption is all or nothing, either you have the secret key and can find out the message, or you don't have the secret key so you can't. With functional encryption we broaden the possibilities of what is communicated between senders in an encryption scheme. We start with a definition and then show the formal construction.

Definition 3 (Correctness). A Functional Encryption Scheme Π is **correct** if for $m \in M$, some predetermined function f with M as its domain, and appropriate public key and evaluation key $(pk, ek) \in K$ from Π 's key generation algorithm:

$$Dec_{ek}(Enc_{pk}(m)) = f(m)$$

It's easy to see that this definition encapsulates the older definition of correctness by making f the identity function $f(m) = m$, but this syntax covers many other cryptographic primitives as well like Attribute Based Encryption and Identity Based Encryption. To see how these primitives are sub cases of Functional Encryption. Lets formalize our idea of a Functional Encryption Scheme.

To define a Functional Encryption Scheme, we must first define a way of describing what a cipher text can be decrypted to.

Think of something better than case space, it's confusing with the notation for a cipher space. The paper calls it a key space K but that's also confusing. Change later, keep in mind now.

Definition 4 (Functionality). *Given a case space $C_{ase} \cup \{\epsilon\}$, message space M we define the functionality F to be*

$$F : C_{ase} \times M \rightarrow M$$

Functionality describes what the possible outputs are. In public key encryption, knowing the secret key sk allows for the message to be read in full, but without the secret key, only the length of the message can be discerned from the cipher text. To write this in the syntax of a functionality we define

$$F(c, m) = \begin{cases} x & \text{if } c = 1 \\ \text{length}(x) & \text{if } c = \epsilon \end{cases}$$

The only functionality of public key encryption is fully decoding the message so this is our primary case ($c = 1$), however we also account for the information learned without the public key which is an unavoidable rather than built in case ($c = \epsilon$).

Definition 5 (Functional Encryption Scheme). *A Functional Encryption Scheme Π is defined to be the following four algorithms:*

$$\text{Setup} : \mathbb{Z} \rightarrow K \times K$$

Defined for $\lambda \in \mathbb{Z}$; to be $\text{setup}(\lambda) \rightarrow (pk, mk)$, generates a public key and master key

$$\text{Gen} : K \times C \rightarrow K$$

Defined for $c \in C_{ase}, mk \in K$; to be $\text{Gen}(mk, c) \rightarrow sk$ which is kept secret and is the secret key of functionality c .

$$\text{Enc} : K \times M \rightarrow C_{ipher}$$

Defined: For $pk \in K$ and $m \in M$; to be $\text{Enc}_{pk}(m) \rightarrow c$

$$\text{Dec} : K \times C_{ipher} \rightarrow M$$

Defined for $ek \in K$ and $c \in C_{ipher}$; to be $\text{Dec}(k, c) \rightarrow n$ where $n = F(k, m)$ for some functionality F .

If the notion of a Functionality was confusing before, the use of it in generating the secret key should make it clear.

Suppose Alice wants to store some text files ($\{m_1, m_2, \dots, m_n\}$) on Bob's Cloud Storage to retrieve at some later date. Bob, owner of Bob's Cloud Storage, has a strong dislike for cats and doesn't want anything stored on his cloud with the word "cat" in it. Alice and Bob contact Janet, an impartial third party, and share their desired functionalities: Alice wants to be able to fully decrypt (i.e. identity functionality), Bob wants to be able to check if the word "cat" is in anything stored on his server (we call this function checking for the word "cat" f). Janet runs $\text{Setup}(\lambda) \rightarrow (mk, pk)$ and publishes the public key, and also runs $\text{Keygen}(mk, 0) \rightarrow sk_{id}$ and $\text{Keygen}(mk, 1) \rightarrow sk_f$ for functionality:

$$F(c, m) = \begin{cases} f(m) & \text{if } k = 0 \\ id(m) & \text{if } k = 1 \end{cases}$$

Then Janet securely sends sk_{id} to Alice and sk_f to Bob. Now Alice encrypts her text files $\text{Enc}_{pk}(m_1) \rightarrow c_1, \text{Enc}_{pk}(m_2) \rightarrow c_2, \dots, \text{Enc}_{pk}(m_n) \rightarrow c_n$, and sends $\{c_1, c_2, \dots, c_n\}$ to Bob for storage. Bob then runs $\text{Dec}_{sk_f}(c_1) \rightarrow b_1, \dots, \text{Dec}_{sk_f}(c_n) \rightarrow b_n$ and allows storage of all c_i that indicate the word "cat" is not being stored. This interaction is illustrated in the diagram below:

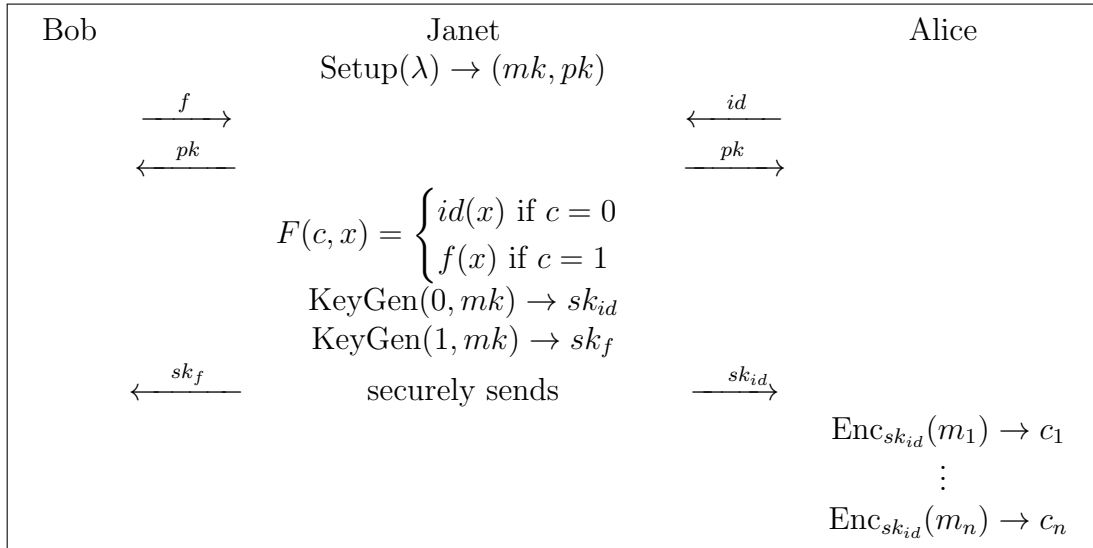


Table 1.2: Public Key Encryption Protocol

It should be noted that the impartial 3rd party does not need to be fully trusted since Janet doesn't have any of Alice's cipher text's to decrypt, but it is important that Janet is impartial since otherwise she could collaborate with Alice to store a Cat Encyclopedia on Bob's Servers or could collaborate with Bob to learn more about what Alice is storing than what Alice and Bob have agreed upon.

Research into Functional Encryption is currently at early stages, later we will describe the details of how it works, and where it's currently limited. Even at such

an early stage Cryptographic research, Functional Encryption has proven to be a powerful tool giving us a variety of functionalities with only small alterations of syntax such as Attribute Based Encryption **CITATION NEEDED**, Identity Based Encryption **CITATION NEEDED**, and Multi-Party Input Functional Encryption **CITATION NEEDED** which we will actually work with later on.

1.2 Obfuscation

1.2.1 Black Box Obfuscation

In short, circuit (and program) Obfuscation is a Cryptographic method that seeks to make a circuit unintelligible to anyone looking at only its obfuscation. Black Box Obfuscation was first defined by **(INSERT CITATION)**. The goal of Black Box Obfuscation is that a circuit that has been Black Box Obfuscated should reveal no more about its inner workings than a table of inputs and outputs of the that circuit (we call this table an Oracle since).

Definition 6. *Given any circuit C and an Oracle $\mathcal{O}_c(\cdot)$ with the same functionality of C , an obfuscator $\mathcal{O}(\cdot)$, and any adversary \mathcal{A} . We say \mathcal{O} is a Black Box obfuscator if \mathcal{A} given access to $\mathcal{O}(C)(\cdot)$ can determine just as much about the inner workings of C as if \mathcal{A} had been given access to $\mathcal{O}_c(\cdot)$.*

FIND A BETTER DEFINITION OF BLACK BOX OBFUSCATION

While Black Box Obfuscation has been proven impossible for general circuits and programs **(insert citation and perhaps some more details as to what exactly has been proven impossible)**

Let's take a moment to talk about why Cryptographers would want Black Box Obfuscation as a tool. Currently Public Key Encryption relies on expensive computations, (For Example in RSA the private keys are just inverses of public keys in some group large enough for this to be difficult to compute). However, private key encryption is much more efficient since it's just running some sort of permutation on the message using the secret key and then descrambling the cipher text with the same secret key. Using Black Box Obfuscation it would be possible to obfuscate a private key encryption function with a secret key sk baked in $\mathcal{O}(\text{Enc}_{sk}(\cdot) \rightarrow \text{Enc}(\cdot))$ where $\text{Enc}(\cdot)$ can be made public without risk of anyone learning sk keeping the ability to decrypt in the hands of those who started off with the secret key. Thus Black Box Obfuscation would make efficient private key encryption into efficient public key encryption.

1.2.2 Indistinguishability Obfuscation

While Black Box Obfuscation was proved impossible **CITATION**, Cryptographers weakened their definition of Obfuscation to try and see what *is* possible in the field of obfuscation. This led to a new notion of Obfuscation called Indistinguishability Obfuscation.

Definition 7 (Indistinguishability Obfuscation). *Given circuits C_0, C_1 where $|C_0| \approx |C_1|$, $C_0(x) = C_1(x)$ for all valid x , and an obfuscater $i\mathcal{O}(\cdot)$; we say that $i\mathcal{O}$ is an **indistinguishability obfuscater** if for all Distinguishers \mathcal{D}*

$$\Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1] \leq \text{negl}$$

This definition can be a little confusing to understand. In short, Indistinguishability Obfuscation guarantees that two circuits with the same functionality are indistinguishable from one another once run through an indistinguishability obfuscater.

What can be more confusing is why this would be useful since the two circuits are functionally the same. The most simple usage is in removing water marks from programs. Suppose Dan buys a fancy piece of software called Macrosoft Word for his company. Macrosoft is worried about their software being pirated so they put a watermark in Dan's copy of Macrosoft Word that doesn't change the functionality of his copy of the program, indicating that this is Dan's copy. Suppose Dan wants to make Macrosoft Word free for everyone and decides to post it on a torrenting website. If Dan posts his copy as is, Macrosoft and their Copyright lawyers will be able to see that it was Dan who illegally shared his copy of their software. Instead, if Dan first runs his copy of Macrosoft word through an Indistinguishability Obfuscater and post that, Macrosoft and their lawyers will be unable to tell if it was Dan's copy that was posted, or another customer's.

This might seem like a weak definition, but Indistinguishability Obfuscation has become a powerful cryptographic tool, and has proven to be as good as the best possible obfuscation, the proof is short and eloquent so we will show it here but it was originally formulated by (**INSERT CITATION HERE**)

Proof. Let $i\mathcal{O}(\cdot)$ be an indistinguishability obfuscater, $\mathcal{O}(\cdot)$ the best obfuscater possible, and C a circuit. Then by definition of Indistinguishability Obfuscation, $i\mathcal{O}(C)$ is indistinguishable from $i\mathcal{O}(\mathcal{O}(C))$. Thus Indistinguishability Obfuscation is at least as good as any other type of Obfuscation. \square

Because of this, Cryptographers often treat Indistinguishability Obfuscation the same as Black Box Obfuscation. This could lead to issues in the future and there is lots of room for work to be done in quantifying degrees of Obfuscation.

1.3 Bilinear Maps

Lastly we give a brief introduction to Bilinear maps, which we will later generalize to Multilinear maps.

1.3.1 Definition

Definition 8. Bilinear Maps Let G_1, G_2, G_3 be cyclic groups of prime order p . Then we say a map $e : G_1 \times G_2 \rightarrow G_3$ is bilinear if

1. For all $g_1 \in G_1, g_2 \in G_2$, and $\alpha \in \mathbb{Z}_p$, $e(\alpha \cdot g_1, g_2) = e(g_1, \alpha \cdot g_2) = \alpha \cdot e(g_1, g_2)$
2. For generators $g_1 \in G_1$ and $g_2 \in G_2$, $e(g_1, g_2) = g_3$ a generator of G_3 .

1.4 Intuition

The most accessible example of Bilinear Map is in Tripartite Diffie-Hellman Key Exchange. Here the Bilinear Map $e : G_1 \times G_1 \rightarrow G_2$ is defined $e(g_1^a, g_1^b) \rightarrow g_2^{a \cdot b}$ for generators $g_1 \in G_1$ and $g_2 \in G_2$. Here we can think of a and b as secrets encoded by making them a power of a generator. Using the map e we can multiply secrets and by multiplying encodings $g^a \cdot g^b = g^{a+b}$ we can also add secret elements.

Suppose Alice, Bob and Carol wish to agree upon a single secret key for the three of them. To do so Alice would run $\text{InstGen}(\lambda, 2) \rightarrow (\mathbf{params}, g_1)$ where g_1 generates G_1 . Then she would choose a secret $a \in \mathbb{Z}_p$ and broadcast $(e \in \mathbf{params}, g_1, pk_a = g_1^a)$. Then Bob would choose $b \in \mathbb{Z}_p$ and broadcast $pk_b = g_1^b$ and Alice would do the same broadcasting $pk_c = g_1^c$. For each party to learn the secret key they would need to run:

$$\text{Alice: } e(pk_b, pk_c)^a = e(g_1^b, g_1^c)^a = (g_2^{b \cdot c})^a = g_2^{a \cdot b \cdot c} = sk$$

$$\text{Bob: } e(pk_a, pk_c)^b = g_2^{a \cdot b \cdot c} = sk$$

$$\text{Carol: } e(pk_a, pk_b)^c = g_2^{a \cdot b \cdot c} = sk$$

The important thing to understand is that this 2-Linear Map makes it possible for 3 parties to share a secret without risk of an eavesdropper also being able to know the secret. To generalize, a k -multilinear map allows us to encode k secrets easily, so if each party involved in the exchange knows a secret this allows for $k + 1$ non interactive key exchange.

Chapter 2

Multi-Linear Maps

2.1 Cryptographic Multilinear Maps

Before we can construct Indistinguishability Obfuscation or Functional Encryption, we must understand a central tool in both called Multilinear Maps.

Definition 9. *Cryptographic Multilinear Map* Let $G_1, G_2, \dots, G_k, G_T$ be cyclic groups each of the same prime order p with generators g_1, g_2, \dots, g_k respectively.. Then we say a map $e : G_1 \times G_2 \times \dots \times G_k \rightarrow G_T$ is a k -multilinear map if

1. For $g_i \in G_i \forall i \in [K]$ where $[K]$ is the set $\{1, 2, \dots, k\}$, and $\alpha \in \mathbb{Z}_p$ then $e(g_1, g_2, \dots, \alpha \cdot g_\ell, \dots, g_k) = \alpha \cdot e(g_1, g_2, \dots, g_\ell, \dots, g_k)$
2. if g_i generates G_i then $e(g_1, g_2, \dots, g_k)$ generates G_T . We say that e is **non-degenerate** if this property holds.

This definition is just a generalization of the one we gave for Bilinear Maps. While we have efficient constructions of Bilinear Maps, we don't have such efficiency with generalized Multilinear Maps **CONFIRM THIS STATEMENT AND GET A CITATION**.

2.2 Graded Encoding Schemes

2.2.1 Definition

Definition 10. *k-Graded Encoding Scheme*

A k -Graded Encoding System consists of a ring R and a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0, 1\}^* \mid \alpha \in R, 0 \leq i \leq k\}$, such that:

1. For every fixed index i , the sets $\{S_i^{(\alpha)} \mid \alpha \in R\}$ are disjoint (meaning they form a partition of $S_i := \cup_{\alpha} S_i^{(\alpha)}$)

2. There is an associative binary operation $'+'$ and a self-inverse unary operation $'-'$ such that $\forall \alpha_1, \alpha_2 \in R$, index $i \leq k$, and $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that

$$u_1 + u_2 \in S_i^{(\alpha_1 + \alpha_2)} \text{ and } -u_1 \in S_i^{(-\alpha_1)}$$

where $\alpha_1 + \alpha_2$ and $-\alpha_1$ are addition and negation in R .

3. There is an associative binary operation $'\times'$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$ every i_1, i_2 with $i_1 + i_2 \leq k$ and every $u_1 \in S_{i_1}^{(\alpha_1)}$ and $u_2 \in S_{i_2}^{(\alpha_2)}$ it holds that

$$u_1 \times u_2 \in S_{i_1 + i_2}^{(\alpha_1 \cdot \alpha_2)}$$

Where $\alpha_1 \cdot \alpha_2$ is multiplication in R , and $i_1 + i_2$ is integer addition.

At first glance, it can be difficult to see how this definition of a Graded Encoding Scheme relates to Multilinear Maps.

2.3 Intuition

2.3.1 Structural Intuition

In practice, cryptographers use Graded Encoding Schemes (also known as Approximate Multilinear Maps) as a means for doing computation on multiple secrets. To give an intuition into the structure of Graded Encoding Schemes we give the simplified definition below.

Definition 11. *k-Graded Encoding Scheme (simplified)*

Given cyclic groups G_1, G_2, \dots, G_k of prime order p . We call the family of bilinear maps

$$e_{i,j} : G_i \times G_j \rightarrow G_{i+j} \text{ for all } 0 \leq i, j \leq k \text{ where } i + j \leq k$$

a simplified Graded Encoding Scheme. For any secret $\alpha \in \mathbb{Z}_p$ and generator g_i of G_i we call g_i^α a level i encoding of α .

In this Encoding Scheme, secrets $\alpha_1, \alpha_2, \dots \in \mathbb{Z}_p$ are initially encoded as they were with bilinear maps, by taking a generator $g_1 \in G_1$ and raising it to the α_1 so that it looks like $g_1^{\alpha_1}$. As stated in the definition we call this a level one encoding.

For a level one encoding of elements, computation on secrets is the same as before, but now if we have $e_{1,1}(g_1^{\alpha_1}, g_1^{\alpha_2}) = g_2^{\alpha_1 \cdot \alpha_2}$ and want to continue to computations on the new secret $\alpha_1 \cdot \alpha_2$ with another secret α_3 we can 'multiply' the level one encoding α_3 with a level one encoding of 1 to get $e_{1,1}(g_1^{\alpha_3}, g_1^1) = g_2^{\alpha_3}$ and continue to do operations as before.

So using this scheme, we can do computations on multiple secrets as long as there are at most k multiplications.

2.3.2 Encodings

In our simplified definition, encodings were of the form g_i^α . In practice cryptographers use sets (i.e. the S_i from the full definition of graded encoding schemes) that allow for multiple different encodings of the same secret at any level.

2.4 Candidate Groups/Quotient Rings/Fields

Chapter 3

Indistinguishability Obfuscation

3.1 Notes

- IO scheme for (polynomial size) NC1 circuits is built using some candidate graded encoding schemes. The latter is an algebraic structure, introduced by Garg et al. [35], that enables homomorphically evaluating certain polynomials over encoded ring elements and testing whether the output is zero.
- a bootstrapping theorem transforms an IO scheme for NC1 into one for P/poly, assuming the LWE assumption

3.2 Construction

3.3 Usage, Limitations, and Goals

Chapter 4

Multi-Party Input Functional Encryption

4.1 Scheme

4.2 Construction

4.3 Limitations and Goals

Chapter 5

A Brief Introduction to the 5-GenC library

5.1 The DSL

5.2 Circuits and Branching Programs

5.3 Base and MMaps

Chapter 6

Experiments

6.1 Comparison Circuit

6.2 Runtime Evaluation

Chapter 7

Conclusion

References

- Angel, E. (2000). *Interactive Computer Graphics : A Top-Down Approach with OpenGL*. Boston, MA: Addison Wesley Longman.
- Angel, E. (2001a). *Batch-file Computer Graphics : A Bottom-Up Approach with QuickTime*. Boston, MA: Wesley Addison Longman.
- Angel, E. (2001b). *test second book by angel*. Boston, MA: Wesley Addison Longman.
- Deussen, O., & Strothotte, T. (2000). Computer-generated pen-and-ink illustration of trees. *"Proceedings of" SIGGRAPH 2000*, (pp. 13–18).
- Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (1997). *Hypermedia Image Processing Reference*. New York, NY: John Wiley & Sons.
- Gooch, B., & Gooch, A. (2001a). *Non-Photorealistic Rendering*. Natick, Massachusetts: A K Peters.
- Gooch, B., & Gooch, A. (2001b). *Test second book by gooches*. Natick, Massachusetts: A K Peters.
- Hertzmann, A., & Zorin, D. (2000). Illustrating smooth surfaces. *Proceedings of SIGGRAPH 2000*, 5(17), 517–526.
- Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Englewood Cliffs, New Jersey: Prentice-Hall.
- Molina, S. T., & Borkovec, T. D. (1994). The Penn State worry questionnaire: Psychometric properties and associated characteristics. In G. C. L. Davey, & F. Tallis (Eds.), *Worrying: Perspectives on theory, assessment and treatment*, (pp. 265–283). New York: Wiley.
- Noble, S. G. (2002). *Turning images into simple line-art*. Undergraduate thesis, Reed College.
- Reed College (2007). Latex your document. <http://web.reed.edu/cis/help/LaTeX/index.html>
- Russ, J. C. (1995). *The Image Processing Handbook, Second Edition*. Boca Raton, Florida: CRC Press.

- Salisbury, M. P., Wong, M. T., Hughes, J. F., & Salesin, D. H. (1997). Orientable textures for image-based pen-and-ink illustration. *“Proceedings of” SIGGRAPH 97*, (pp. 401–406).
- Savitch, W. (2001). *JAVA: An Introduction to Computer Science & Programming*. Upper Saddle River, New Jersey: Prentice Hall.
- Wong, E. (1999). *Artistic Rendering of Portrait Photographs*. Master’s thesis, Cornell University.