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

Chapter 1

Background

1.1 Encryption

In plain English, 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 by obscurity is that as soon as a method of obscurity 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, and the best keys are ones that are chosen randomly, since they are nearly impossible to guess. In defining an encryption scheme it is important to note that there exists a keyspace K which is the set of all valid keys, a message space M made up of all valid messages, and a cipher space C the set of all valid cipher-texts (encryptions of messages).

We define an encryption scheme Π to be the following three functions:

$$\text{Gen} : \mathbb{Z} \to K \times K$$

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

$$\text{Enc} : K \times M \to C$$

Defined to be for key $pk \in K$ and message $m \in M$ $\text{Enc}_p k(m) \to c$ for some cipher text $c \in C$

$$\text{Dec} : K \times C \to M$$

Defined to be for key $sk \in K$ and cipher text $c \in C$ $\text{Dec}_s k(c) \to m$ for some message $m \in M$

It is important to note that 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 1 (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 and wants to send Bob a secret message m. To do this Bob would have to run $\text{Gen}(n) \to pk, sk$ and then send pk to Alice. Then Alice gets $c :=\text{Enc}_{pk}(m)$ and sends c over to Bob. Finally Bob gets $m' :=\text{Dec}_{sk}(c)$. 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.

insert sick diagram of Alice interacting with Bob

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 2 (Correctness)**.** *A Functional Encryption Scheme* Π *is* ***correct*** *if for* $m \in M$*, some predetermined function* f *with* M *as it's domain, and appropriate* $(pk, ek) \in K$ *generated by* 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 3 (Functionality)**.** *Given a case space $C_ase \cup \{\epsilon\}$, message space M we define the functionality F to be*

$$F : C_ase \times M \to M$$

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

$$Setup : \mathbb{Z} \to K \times K$$

Defined: For $\lambda \in \mathbb{Z}$, setup(λ) $\to (pk, mk)$, generates a public key and master key

$$Gen : K \times C \to K$$

Defined: For $c \in C_{ase}, mk \in K$, $Gen(mk, c) \to sk$ which is kept secret and is the secret key of functionality c.

$$Enc : K \times M \to C_{ipher}$$

Defined: For $pk \in K$ and $m \in M, Enc_{pk}(m) \to c$

$$Dec : K \times C_{ipher} \to M$$

Defined: For $ek \in K$ and $c \in C_{ipher}, Dec(k, c) \to n$ 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 ($\{m_1, m_2, \cdots, m_n\}$) 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, 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 Setup(λ) $\to (mk, pk)$ and

publishes the public key, and also runs Keygen$(mk, 0) \to sk_{id}$ and Keygen$(mk, 1) \to sk_f$ 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 $\text{Enc}_{pk}(m_1) \to c_1$,$\text{Enc}_{pk}(m_2 \to c_2, \cdots$,$\text{Enc}_{pk}(m_n) \to c_n$,, and sends $\{c_1, c_2, \cdots, c_n\}$ to Bob for storage. Bob then runs $Dec_{sk_f}(c_1) \to b_1, \cdots, Dec_{sk_f}(c_n) \to b_n$ and allows storage of all c_i that indicate the word "cat" is not being stored. This interaction is illustrated in the diagram below:

insert sick diagram of Alice Bob and Janet executing the exchange discussed above!

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 should 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.

Functional Encryption is at the early stages of development now, but is an extremely powerful tool. From Functional Encryption we can easily describe variations like Attribute Based Encryption, Identity Based Encryption, and Multi Input Functional Encryption.

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 it's inner workings than a table of inputs and outputs of the that circuit (we call this table an Oracle since).

Definition 5. *Given any circuit C and an Oracle $\mathcal{O}_c(\cdot)$ with the same functionality of C, an obfuscater $\mathcal{O}(\cdot)$, and any adversary $\mathcal{A}$. We say $\mathcal{O}$ is a Black Box obfuscater 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, since the private keys are just inverses of public keys in some group so the groups and the keys have to be extremely large, however private key encryption is much more efficient since it's just running some sort of permutation on the message using the key and then descrambling the cipher text with the same key. Using Black Box Obfuscation if would be possible to obfuscate an encryption function with a secret key, sk, baked in $\mathcal{O}(Enc_{sk}(\cdot) \rightarrow Enc(\cdot)$ where $Enc(\cdot)$ 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.

1.2.2 Indistinguishability Obfuscation

While Black Box Obfuscation was proved impossible, 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 6 (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 $\mathcal{D}$*

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

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 obvious 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. 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 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 $\mathcal{O}(C)$. Thus Indistinguishability Obfuscation is at least as good as any other type of Obfuscation. □

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 levels of Obfuscation.

Chapter 2

Multi-Linear Maps

2.1 Definition

2.2 Intuition

2.3 Construction Outline

2.4 Candidate Goups/Quotient Rings/Fields

Chapter 3

Indistinguishability Obfuscation

3.1 Definition

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.