

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
1.3.2 Intuition	9
1.3.3 Hardness Assumption	9
Chapter 2: Multi-Linear Maps	11
2.1 Cryptographic Multilinear Maps	11
2.1.1 Dream Definition	11
2.1.2 Hardness Assumption	12
2.2 Graded Encoding Schemes	12
2.2.1 Intuition	12
2.2.2 Encodings	13
2.2.3 Public and Private Parameters/Zero Test	13
2.2.4 Definition	13
2.2.5 Procedures	14
2.2.6 Hardness Assumptions	15
Chapter 3: Indistinguishability Obfuscation	17
3.1 Circuit Representation	17
3.1.1 Branching Programs	17
3.1.2 Barrington's Theorem	18
3.2 Approaching Obfuscation for NC^1	19
3.2.1 Kilian's Protocol	19
3.2.2 Encoded Random Branching Programs	19
3.3 Construction	19
3.4 Usage, Limitations, and Goals	19

Chapter 4: Functional Encryption	21
4.1 Definition	21
4.2 Construction	21
Chapter 5: A Brief Introduction to the 5-GenC library	23
5.1 The DSL	23
5.2 Circuits and Branching Programs	23
5.3 Base and MMaps	23
Chapter 6: Experiments	25
6.1 Comparison Circuit	25
6.2 Runtime Evaluation	25
Chapter 7: Conclusion	27
References	29

Introduction

Chapter 1

Background

1.1 Encryption

Encryption is 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 also 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 and chosen randomly over some distribution (often uniformly), so it is infeasible for an adversary to check every possible key or guess the right key using information about communicating parties.

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 in the public key setting 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.

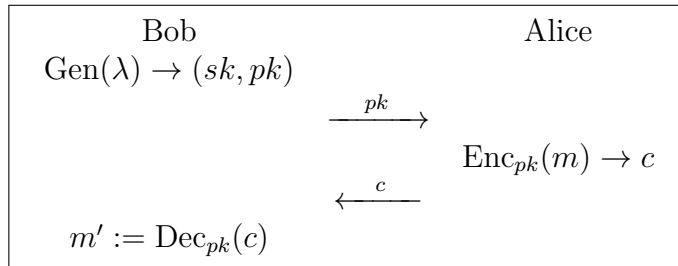


Table 1.1: Public Key Encryption Protocol

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 (ABE) and identity based encryption (IBE). To see how these primitives are sub cases of functional encryption lets formalize our idea of a functional encryption scheme.

Definition 4 (Functional Encryption Scheme). *Given a message space M , a cipher space C , a family of functions \mathcal{F} , and a key space K , 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 $\text{setup}(\lambda) \rightarrow (PP, MSK)$, generates a public parameter and master secret key both in K

$$\text{Gen} : K \times \mathcal{F} \rightarrow K$$

Defined for $f \in \mathcal{F}, MSK \in K$; to be $\text{Gen}(mk, c) \rightarrow SK_f$ which is kept secret and is the secret key associated with the functionality of f .

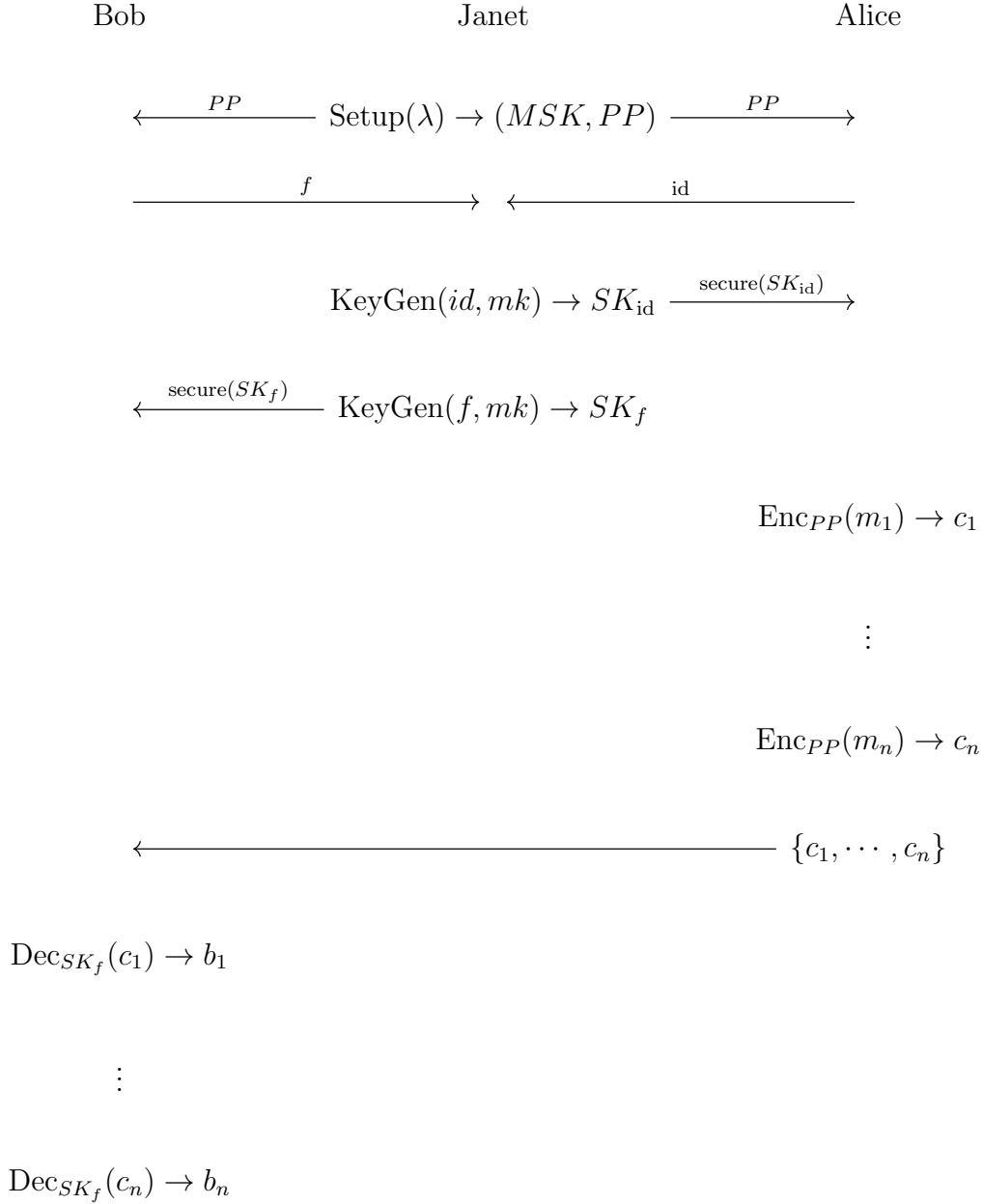
$$\text{Enc} : K \times M \rightarrow C$$

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

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

Defined for $SK_f \in K$ and $c \in C$; to be $\text{Dec}(k, c) \rightarrow m'$ if the scheme is correct then $m' = f(m)$ for some previously encrypted m and some $f \in \mathcal{F}$.

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 $\text{Setup}(\lambda) \rightarrow (PP, MSK)$ and publishes the public parameter, and also runs $\text{Keygen}(MSK, id) \rightarrow SK_{id}$ and $\text{Keygen}(mk, f) \rightarrow SK_f$. 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:



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 texts 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, identity based encryption, and multi-party input

functional encryption [Boneh et al. (2011)] which we will actually work with later on.

1.2 Obfuscation

1.2.1 Black Box Obfuscation

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 as defined in [Barak et al. (2001)] 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 for the same circuit (we call this table an Oracle).

Definition 5. *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 $\mathcal{O}_c(\cdot)$.*

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 in [Barak et al. (2001)], 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.

Before defining indistinguishability obfuscation it is worth noting that a distinguisher is any polynomial time algorithm that can differentiate two things. For example, a distinguisher for apples and oranges might look at the color of what it has been given and output Orange if the color is orange or Apple if the color of what it's been given is red/green/yellow, however another distinguisher for apples and oranges might just flip a coin and output Apple if the coin comes up heads and Orange otherwise, the second distinguisher would probably not be a very good distinguisher. Distinguishers are only defined on the two things that they distinguish, apples or oranges, green or blue, paper or not paper.

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}

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

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 we require the two circuits to be 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 the best possible obfuscation, the proof is short and eloquent so we will show it here but it was originally formulated by [Goldwasser & Rothblum (2007)].

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. While this could lead to issues in the future and there is lots of room for work to be done in quantifying degrees of Obfuscation, we will fall back on the assumption that the best possible obfuscation is good enough for our purposes.

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 7. *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 $g_1 \in G_1, g_2 \in G_2$, and $g_3 \in G_3$, $e(g_1, g_2) = g_3$

1.3.2 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 $g_2 \in G_2$. We can think of a, b as secrets and g_1^a, g_1^b as their encodings in G_1 . Using the map e we can multiply secrets and by multiplying encodings $g^a \cdot g^b = g^{a+b}$ we can also add secrets.

Suppose Alice, Bob and Carol wish to agree upon a single secret key for the three of them. To do so Alice would run an instance generation function that outputs a description of the group G_1 , a generator g_1 and a bilinear map e as we defined above. 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:

$$\begin{aligned} \text{Alice: } e(pk_b, pk_c)^a &= e(g_1^b, g_1^c)^a = (g_2^{b \cdot c})^a = g_2^{a \cdot b \cdot c} = sk \\ \text{Bob: } e(pk_a, pk_c)^b &= g_2^{a \cdot b \cdot c} = sk \\ \text{Carol: } e(pk_a, pk_b)^c &= g_2^{a \cdot b \cdot c} = sk \end{aligned}$$

The important thing to understand is that this bilinear 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.

1.3.3 Hardness Assumption

In order to talk about how secure it is to run protocols that use bilinear maps (such as Alice, Bob, and Carol's key exchange) we need to formalize what the adversary would need to be able to do in order to compromise security. We call these formalizations *hardness assumptions*.

First we define the following problems.

Definition 8. *Bilinear Computational Diffie-Hellman Problem*

Let G_1, G_2 be cyclic groups of prime order p and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. For $a, b, c \in \mathbb{Z}_p$ chosen uniformly and g_1, g_2 generators of G_1, G_2 respectively. The Computational Diffie-Hellman problem is: given e, g_1, g_1^a, g_1^b and g_2^c ; compute

$$e(g_1, g_1^{a \cdot b \cdot c}) = g_2^{a \cdot b \cdot c}$$

The computational Diffie-Hellman problem might seem a bit confusing, but it is pretty much just a formalization of what we realized after discussing tripartite key exchange: multiplying two secrets is easy using bilinear maps, but that multiplying by a third secret is difficult. Thus we have

Hardness Assumption 1. *Bilinear CDD hardness assumption*

The bilinear computational Diffie-Hellman problem is hard.

Another hardness assumption that we need to make is the bilinear discrete logarithm problem defined below.

Definition 9. *Bilinear Discrete Logarithm* Let G_1, G_2 be cyclic groups of prime order p and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. For $a \in \mathbb{Z}_p$ chosen uniformly, and generators g_1, g_2 of G_1, G_2 respectively, the bilinear discrete logarithm problem is: given g_1, g_1^a, G_1, G_2 , and e find a .

Note that while the above problem may seem easy, the encoding $g_1^a = g$ just looks like some element of G_1 , and since g_1 is a generator of G_1 , *any* element of G_1 can be written as $g_1^\beta \forall \beta \in \mathbb{Z}_p$. We want it to be difficult to get a secret out of its encoding, so we also rely on the following assumption.

Hardness Assumption 2. *Bilinear DL hardness assumption*

The bilinear discrete logarithm problem is hard

There are arguments about if these are *good* assumptions to be making, but these arguments are outside the scope of this thesis. It is enough for us that so far cryptographers and mathematicians have been unable to give efficient solutions to solving these problems in general cases.

Chapter 2

Multi-Linear Maps

2.1 Cryptographic Multilinear Maps

2.1.1 Dream Definition

In order to construct indistinguishability obfuscation or functional encryption, we need to be able to do computations on encoded secrets securely. Specifically, we want to be able to encode, add, multiply and negate secrets. In the previous chapter, we saw that bilinear maps allow us to multiply two encoded secrets securely, but if we want to be able to securely do more complex computations involving multiple multiplications, or sums of products, or products of sums, we need a new cryptographic tool that allows for greater functionality.

Definition 10. *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 $e(g_1, g_2, \dots, g_\ell^\alpha, \dots, g_k) = e(g_1, g_2, \dots, g_\ell, \dots, g_k)^\alpha$
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.

As before we can think of $\alpha \cdot g_i$ to be g_i^α for $g_i \in G_i$ and $\alpha \in \mathbb{Z}_p$. In this setting, we would use the map e to do k multiplications of encodings in our secure computation. For example for $a, b, c, d, e, f, h \in \mathbb{Z}_p$ and generators $g_1 \in G_1, g_2 \in G_2, g_3 \in G_3$ and $g_4 \in G_4$ we could compute $(a + b) \cdot (c + d) \cdot (e + f + h)$ with the 3-linear map $e : G_1 \times G_2 \times G_3 \rightarrow G_4$ with

$$\begin{aligned} e(g_1^a \cdot g_1^b, g_2^c \cdot g_2^d, g_3^e \cdot g_3^f \cdot g_3^h) &= e(g_1^{a+b}, g_2^{c+d}, g_3^{e+f+h}) \\ &= e(g_1, g_2, g_3)^{(a+b)(c+d)(e+f+h)} \\ &= g_4^{(a+b)(c+d)(e+f+h)} \end{aligned}$$

2.1.2 Hardness Assumption

As before we would like to formalize what is being assumed to be difficult in order to be able to discuss the security of using multilinear maps to do computations on secrets in the clear. The first is just an extension of the bilinear discrete logarithm problem.

Definition 11. *Multilinear Discrete-Log* The same as for the Bilinear DL problem except an adversary is also given the index $i < k$ to know which group they are working in.

and the other is

Definition 12. *Multilinear Decisional Diffie-Hellman Problem*

Let $\alpha, \alpha_1, \alpha_2, \dots, \alpha_k, \alpha_{k+1} \in \mathbb{Z}_p$ be chosen uniformly. The MDDH problem is: given $G_1, G_2, \dots, G_k, G_T$ cyclic groups each of the same prime order p with generators g_1, g_2, \dots, g_k respectively, $g_1^{\alpha_1}, g_2^{\alpha_2}, \dots, g_i^{\alpha_i}, g_i^{\alpha_{i+1}}, \dots, g_k^{\alpha_{k+1}}$ and multilinear map e ; distinguish

$$\prod_{j=1}^{k+1} \alpha_j \cdot e(g_1, g_2, \dots, g_k) \quad \text{from} \quad \alpha \cdot e(g_1, g_2, \dots, g_k)$$

2.2 Graded Encoding Schemes

2.2.1 Intuition

To help build an intuition for how Graded Encoding Schemes work, we begin with a simplified definition, and then we'll build up to the actual definition from there.

Definition 13. *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

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

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 where elements of \mathbb{Z}_p are called level zero encodings.

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 $e_{1,1}(g_1^{\alpha_1}, g_1^{\alpha_2}) = g_2^{\alpha_1 \cdot \alpha_2}$ for generator g_2 of G_2 and want to continue to do 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.

2.2.2 Encodings

As we said in the beginning of this section, we gave a simplified definition of graded encoding schemes in order to give intuition into the tiered structure of the scheme. In our simplified definition, encodings are unique and of the form g_i^α . In practice there are sets of valid encodings for a secret $\alpha \in \mathbb{Z}_p$ at any level, we denote the set of valid encodings of α at level i : $S_i^{(\alpha)}$.

Definition 14.

Let R be a cyclotomic ring, $R_q = R/qR$ for prime q , $g \in R_q$ be small, $z \in R_q$ chosen randomly (so it won't be small), and $I = \langle g \rangle$ the principal prime ideal of g . We define a valid level i encoding of $\alpha \in R_q \cong \mathbb{Z}_p$ to be anything in the set

$$S_i^\alpha = \left\{ \left[\frac{a}{z^i} \right]_q \mid a \in \alpha + I \text{ and } a \text{ is small} \right\}$$

For more details into the definition of short, long, and why it is safe to assume z is invertible in this ring, see **CITATION**, we will instead show how this encoding allows us to achieve multilinear map like functionality.

Let $a \in S_0^\alpha$ and $b \in S_0^\beta$. Then

$$\left[\frac{a}{z^i} \right]_q \cdot \left[\frac{b}{z^j} \right]_q = \left[\frac{a \cdot b}{z^{i+j}} \right]_q$$

So multiplication of a level i encoding with a level j encoding gives a level $i + j$ encoding of their product. Also

$$\left[\frac{a}{z^i} \right]_q + \left[\frac{b}{z^i} \right]_q = \left[\frac{a + b}{z^i} \right]_q$$

So this encoding gives us the properties we were looking for in a graded encoding scheme.

2.2.3 Public and Private Parameters/Zero Test

2.2.4 Definition

If this is your first time reading this paper, it is recommended to read the following definition, but not expect to understand much of it, then after reading the intuition and construction section, reread the formal definition and it should be much clearer.

Definition 15. 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 $S_v := \cup_{\alpha} S_v^{(\alpha)}$)

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.

2.2.5 Procedures

- **Instance Generation:**

- Inputs: λ , our security parameter, and k the maximum number of multiplication our computation might need (note how this will end up being the degree of our k -graded encoding scheme)
- Outputs: \mathcal{S} , the description of a k graded encoding scheme as described above, and p_{zt} a level k zero-test parameter.

- **Ring Sampler:** takes \mathcal{S} as input and outputs a level-zero encoding $a \in S_0^{(\alpha)}$ for uniform $\alpha \in R$. The details of how this procedure is constructed is important, but outside the scope of this thesis. What is important to understand is that because this gives an encoding of a uniform *plaintext* element of R without indicating what that plaintext element is, there is a negligible probability that an adversary would be able to get and recognize a plaintext encoding of 1 or something else that would compromise the security of the multilinear map. However this does give companions of the party that ran the Instance Generation the ability to participate in k -nary key exchange by running the Ring Sampler, saving their plaintext encoding, and then publicizing a higher level encoding.

- **Encoding:** takes as input \mathcal{S} , the level zero encoding $a \in S_0^{(\alpha)}$, and i and then outputs $u \in S_i^{(\alpha)}$, a level i encoding of a

- **Addition, negation, multiplication:** all as we've already described

- **Zero-test:** takes as inputs \mathcal{S} and u , outputs 1 if $u \in S_k^{(0)}$ and 0 otherwise. When used with subtraction this gives us the ability to do equality checking. Note that the zero test only works with level k encodings meaning we can only get information about the encoded elements at the last step of a computation. This is where multilinear maps and graded encoding schemes are different from encrypted computation methods like fully homomorphic encryption. With

graded encoding schemes we can only check if the *result* of a computation is what we expect, while with FHE any and all steps of the computation can be decrypted in full even the initial values. This is why graded encoding schemes are a crucial tool for functional encryption where we want this exact property.

- **Extraction:????**

2.2.6 Hardness Assumptions

The hardness assumptions we have discussed up to this point are pretty standard and well studied, but with the creation of new cryptographic tools comes new hardness assumptions. For this graded encoding scheme a new hardness assumption had to be formulated and while it is inspired by DDH.

Definition 16. *Graded DDH*

1. $(\mathcal{S}, p_{zt}) \leftarrow \text{InstGen}(\lambda, k)$
2. For $i = 1, \dots, k + 1$:
3. Choose $\alpha_i \leftarrow \text{samp}(\mathcal{S})$ # get $k + 1$ level-0 encodings
4. Set $u_i \leftarrow \text{encode}(\mathcal{S}, 1, \alpha_i)$ # get level-1 encodings of α_i
5. Set $\tilde{\alpha} = \prod_{i=1}^{k+1} \alpha_i$ # product of $k + 1$ level-0 encodings
6. Choose $\bar{a} \leftarrow \text{samp}(\mathcal{S})$ # get level-0 encoding of random element
7. Set $\tilde{u} \leftarrow \text{encode}(\mathcal{S}, k, \tilde{\alpha})$ # level- k encoding of the product
8. Set $\bar{u} \leftarrow \text{encode}(\mathcal{S}, k, \bar{a})$ # level- k encoding of random element
9. $\mathbf{u} \leftarrow \{\tilde{u}, \bar{u}\}$ chosen uniformly

Given $\mathcal{S}, u_1, u_2, \dots, u_{k+1}$, and \mathbf{u} , determine if \mathbf{u} is the level k encoding of the product of u_1, u_2, \dots, u_{k+1} or the level k encoding of a random element.

As with our other DDH definitions, when we assume this is hard, we are really saying that in a k -graded encoding scheme, it is hard to determine the product of $k + 1$ level one encodings. We won't go much into how *good* this assumption is, we will say that this encoding scheme is far less studied than, say, RSA and so the cryptanalysis as much less extensive. However there has been some cryptanalysis done on graded encoding schemes and overall, it seems that depending on what type of security is needed, different variations on the scheme can be used to protect against different attacks.

Chapter 3

Indistinguishability Obfuscation

3.1 Circuit Representation

While multilinear maps are a central tool used for the obfuscation of circuits, it is difficult to see how when most of us think of circuits as boolean expressions of the form

$$f(x, y, z) = (x \cdot y) + z$$

which denotes the circuit $(x \text{ and } y) \text{ or } z$. A first thought might be to encode x, y and z with a k -graded encoding scheme, however this wouldn't hide the operations used in f , only the values of the starting parameters. For circuit obfuscation we want it to be hard to determine what f is.

3.1.1 Branching Programs

We want to find another way of representing a circuit C that will allow us to obscure it's inner workings. To do this we work with variations of a data structure called a branching program, which we first give an intuition to how it works with the following example.

For inputs $x_1, x_2, x_3, x_4 \in \{0, 1\}$ a branching comparison program to see if $x_1x_2 > x_3x_4$ is

i	$\text{inp}(i)$	if $\text{inp}(i) = 0$ go to	if $\text{inp}(i) = 1$ go to
1.	1	2	3
2.	3	4	θ
3.	3	I	4
4.	2	θ	5
5.	4	I	θ

In the above program, the first column i gives us the index of which step of the program we are at, the second column $\text{inp}(i)$ gives us which index of the input we are looking at (i.e. $\text{inp}(1)$ indicates we are looking at x_1). The third and fourth columns give us either the next step of the program to go to depending on the value of $\text{inp}(i)$

or a terminal value where I indicates the program output is 1 or θ indicating the terminal output is 0.

With this intuition we formalize a variant of branching programs that uses matrices rather than line numbers to get the same functionality.

Definition 17. (*Oblivious*) *Branching Program*

An oblivious branching program of length- n for ℓ -bit inputs is a sequence

$$BP = ((inp(i), A_{i,0}, A_{i,1}))_{i=1}^n$$

Where the $A_{i,b}$'s are permutation matrices in $\{0,1\}^{5 \times 5}$, and $inp(i) \in [\ell]$ is the input bit position examined at step i of the branching program. The function computed by this branching program is

$$f_{BP, A_0, A_1}(x) := \begin{cases} 1 & \text{if } \prod_{i=1}^n A_{i, x_{inp(i)}} = I \\ 0 & \text{otherwise} \end{cases}$$

As opposed to the example branching program, one of two matrices $A_{i,0}$ or $A_{i,1}$ is chosen based on the bit value of the input indicated by $inp(i)$. Also, note that we can randomize these branching programs in the following way

Definition 18. *Randomized Branching Programs*

Given a branching program $\mathcal{BP} = (inp(i), A_{i,0}, A_{i,1})_{i=1}^n$ and n random 5×5 invertible matrices R_1, R_2, \dots, R_n . A random branching program is the branching program made up of $\tilde{A}_{i,b} = R_{i-1} \cdot A_{i,b} \cdot R_i^{-1}$ of the form

$$\mathcal{RBP} = (inp(i), \tilde{A}_{i,0}, \tilde{A}_{i,1})_{i=1}^n$$

Not only does randomization help to obscure the underlying circuit, it also prevents any sort of attack that involves reordering the matrices, since matrix multiplication isn't commutative.

3.1.2 Barrington's Theorem

Originally branching programs were used for oblivious two party computation, but we can start to use them in the context of obfuscating circuits with the following theorem.

Theorem 1. *Barrington's Theorem*

For any depth d fan-in-2 circuit C , there exists a branching program consisting of at most 4^d permutation matrices $A_{i,b} \in \{0,1\}^{5 \times 5}$ that computes the same function as the circuit C .

Note that in order to keep these branching programs efficient we need our circuit to have logarithmic depth, so this confines us to circuits in Nick's Class (NC^1) although later we will talk about generalizing to polynomial sized circuits.

3.2 Approaching Obfuscation for NC¹

3.2.1 Kilian's Protocol

The first attempt of making indistinguishability obfuscation out for randomized branching programs was Kilian's protocol described below.

Suppose Alice is the obfuscator and Bob is the evaluator. Alice wants to obfuscate circuit $C \in \mathcal{C}$ where \mathcal{C} is the circuit family of all circuits with the same domain and codomain as C . Alice starts with a universal circuit U **say more about this or have a citation ready** that can describe any circuit in \mathcal{C} and x the corresponding input into U such that $U(x, \cdot) = C(\cdot)$. Alice then gets the randomized branching program of

$$U = \mathcal{RBP}_U = (\text{inp}(i), \tilde{A}_{i,0}, \tilde{A}_{i,1})_{i=1}^n$$

Then Alice selects all $\tilde{A}_{i, x_{\text{inp}(i)}}$ for all i with $\text{inp}(i)$ corresponding to a bit in x . Once this is done Alice sends the partially selected random branching program to Bob who selects matrices corresponding to his input y .

It's worth thinking about why this is a good start for obfuscation. When Bob receives the branching program he has no way of knowing which of the two matrices Alice selected, all he sees of the matrices corresponding to her inputs are \tilde{A}_i , just the matrix itself and its order in the product.

Unfortunately, Kilian's protocol alone is broken. However we can further strengthen this scheme using a graded encoding scheme.

3.2.2 Encoded Random Branching Programs

Our first potential fix for Kilian's protocol uses the tool described in chapter 2, graded encoding schemes. As before Alice has a random branching program of a universal circuit, she selects the matrices corresponding to her input x that describes the functionality of C . However, now she generates a n -graded encoding scheme, (where $n = 4^d$ and d is the depth of her original circuit C) and then replaces the entries of each matrix with there level-1 encoding. Now, when Bob multiplies the encoded random branching program together, Alice can send over a level k encoding of the identity matrix so that Bob can zero test to see if the output of his computation is 0 or 1 on his input.

delete this eventually Garg et al. (2013b) Garg et al. (2013a) Carmer et al. (2017)

3.3 Construction

3.4 Usage, Limitations, and Goals

Chapter 4

Functional Encryption

4.1 Definition

Now that we have built up an understanding of multilinear maps and circuit obfuscation, we are ready to talk about functional encryption. Recall the following definition of functional encryption where secret keys are generated for different functionalities.

Definition 19. *Functional Encryption*

A functional encryption scheme for a class of functions $\mathcal{F} = \mathcal{F}(\lambda)$ over message space $\mathcal{M} = \mathcal{M}_\lambda$ consists of four algorithms $\mathcal{FE} = \{ \text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt} \}$ where:

- $\text{Setup}_{\text{FE}}(1^\lambda) \rightarrow (PP, MSK)$ takes security parameter and outputs Public Parameter and Master Secret Key
- $\text{KeyGen}_{\text{FE}}(MSK, f) \rightarrow SK_f$ takes Master Secret Key and a function $f \in \mathcal{F}$ and outputs a corresponding secret key SK_f
- $\text{Encrypt}_{\text{FE}}(PP, m) \rightarrow c$ takes Public Parameter and a message $m \in \mathcal{M}$ and outputs cipher text c
- $\text{Decrypt}_{\text{FE}}(SK_f, c) \rightarrow m'$ takes secret key and cipher text and outputs message m'

We say a functional encryption scheme is correct if

$$\text{Decrypt}(SK_f, \text{Encrypt}(PP, m)) = f(m)$$

4.2 Construction

Here we give a simplified construction of the functional encryption scheme described in the previous section, a more detailed construction using non interactive zero knowledge proofs can be found in Garg et al. (2013b). Here we assume that we have access to a public key encryption scheme with algorithms $(\text{Setup}_{\text{PKE}}, \text{Encrypt}_{\text{PKE}}, \text{Decrypt}_{\text{PKE}})$, then

- $\text{Setup}_{FE}(1^\lambda)$: Takes security parameter λ and generates

$$(PP, MSK) \leftarrow \text{Setup}_{PKE}(1^\lambda)$$

- $\text{KeyGen}_{FE}(MSK, f)$: Outputs an obfuscation $SK_f := \mathcal{O}(P_f(MSK, \cdot))$ for the program $P_{f, MSK}(\cdot) := \text{Decrypt}_{PKE}(MSK, f(\cdot))$
- $\text{Encrypt}_{FE}(PP, m \in \{0, 1\}^n)$: Outputs $c := \text{Encrypt}_{PKE}(PP, m)$
- $\text{Decrypt}_{FE}(SK_f, c)$: Outputs m' by running the obfuscated program SK_f on c .

It might take some rereading of the simplified construction above to understand what's going on. Essentially, our SK_f is an obfuscated program which first fully decrypts a cipher text, then outputs the result of running f on the decrypted message. The idea of fully decrypting the cipher text and then evaluating the function on the decrypted message is why we need obfuscation for this scheme to work since working with an obfuscated program should give no more information than just its input and output behaviors (at-least as we said before this is how indistinguishability obfuscation is often treated).

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

- Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., & Yang, K. (2001). On the (im)possibility of obfuscating programs. In J. Kilian (Ed.), *Advances in Cryptology — CRYPTO 2001*, (pp. 1–18). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Boneh, D., Sahai, A., & Waters, B. (2011). Functional encryption: Definitions and challenges. In Y. Ishai (Ed.), *Theory of Cryptography*, (pp. 253–273). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Carmer, B., Malozemoff, A. J., & Raykova, M. (2017). 5gen-c: Multi-input functional encryption and program obfuscation for arithmetic circuits. Cryptology ePrint Archive, Report 2017/826. <https://eprint.iacr.org/2017/826>.
- Garg, S., Gentry, C., & Halevi, S. (2013a). Candidate multilinear maps from ideal lattices. In T. Johansson, & P. Q. Nguyen (Eds.), *Advances in Cryptology – EUROCRYPT 2013*, (pp. 1–17). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., & Waters, B. (2013b). Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS '13*, (pp. 40–49). Washington, DC, USA: IEEE Computer Society.
- Goldwasser, S., & Rothblum, G. N. (2007). On best-possible obfuscation. In *Proceedings of the 4th Conference on Theory of Cryptography, TCC'07*, (pp. 194–213). Berlin, Heidelberg: Springer-Verlag. <http://dl.acm.org/citation.cfm?id=1760749.1760765>