7-31-2009

Discrete Logarithm over Composite Moduli

Marcus L. Mace
Abilene Christian University

Discrete Logarithm over Composite Moduli

Marcus L. Mace

Adviser: Joshua B. Holden

Mathematical Sciences Technical Report Series
MSTR 09-07

July 31, 2009

Department of Mathematics
Rose-Hulman Institute of Technology
http://www.rose-hulman.edu/math

Fax (812)-877-8333 **Phone (812)-877-8193**

Discrete Logarithm over Composite Moduli

Marcus L. Mace
Abilene Christian University

Faculty Advisor: Joshua Holden
Rose-Hulman Institute of Technology
Mathematics Research Experience for Undergraduates (REU)

Summer 2009

DISCRETE LOGARITHM OVER COMPOSITE MODULI

MARCUS L. MACE

ABSTRACT. In an age of digital information, security is of utmost importance. Many encryption schemes, such as the Diffie-Hellman Key Agreement and RSA Cryptosystem, use a function which maps x to y by

$$y \equiv g^x \pmod{n}$$

for a given n and a generator (or primitive root) g. The inverse of this function - trying to find x from y - is called the discrete logarithm problem. In most cases, n is a prime number. In some cases, however, n may be a composite number. In particular, we will look at when $n = p^b$ for a prime p. We will show different techniques of obtaining graphs of this mapping and then we look to see whether the above mapping for the described n looks like a random map, and, if it does not, observe what we can that would help in solving the discrete logarithm problem.

1. INTRODUCTION

If we consider a function that maps x to y by

$$y \equiv g^x \pmod{n} \tag{1}$$

where $n \in \mathbb{N}$ and g is a primitive root of n, then it is easy to find y given x, g, and n by exponentiating and doing modular arithmetic concurrently. However, if you only know y, g, and n, then finding x turns out to be a very hard problem from a computational perspective. This is known as the *discrete logarithm problem*. The discrete logarithm problem applies most famously to modern day cryptography. Three common examples of encryption using the discrete log problem are the Pohlig-Hellman Cipher, the RSA Cryptosystem, and the Diffie-Hellman Key Agreement. For futher information about these three, refer to pages 253-258 and 260-262 in [8].

Date: July 27, 2009.

2000 *Mathematics Subject Classification.* 11Y99.

Key words and phrases. Discrete Logarithm Problem, Composite Moduli, Functional Graphs.

1.1. **Pohlig-Hellman Cipher.** The Pohlig-Hellman Cipher is the simplest case of the discrete log problem. If Alice wants to send Bob a message M, then they both agree on a large prime, p. They choose a positive integer e less than $p-1$ and find the multiplicative inverse of e modulo $p-1$, that is, an integer d which satisfies $ed \equiv 1 \pmod{p-1}$. Then Alice sends Bob the cipher text $C \equiv M^e \pmod{p}$. To decipher, Bob calculates $C^d \pmod{p}$, which gives him the message M. This works because of Fermat's Little Theorem which states that for any prime p and integer a such that $\gcd(a,p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$. Thus, $C^d \equiv (M^e)^d \equiv M^{1+k(p-1)} \equiv M \cdot M^{k(p-1)} \equiv M \pmod{p}$, recovering the original message M.

1.2. **RSA Cryptosystem.** Ronald Rivest, Adi Shamir and Leonard Adleman proposed a public-key encryption scheme in 1978 aptly named RSA. Alice and Bob agree on two large primes p and q and calculate $n = pq$ to be their modulus. Alice sends Bob a message by looking up Bob's public keys, e and n. If M is the message she wishes to encrypt, she calculates $C \equiv M^e \pmod{n}$, which becomes the cipher text. After Bob receives this message, he then uses his secret key d, the multiplicative inverse of e modulo $\phi(n)$, to decrypt the message by finding $M \equiv C^d \pmod{n}$. The most obvious way to break this scheme would be to find d, Bob's secret key. Unfortunately, this is equivalent to the problem of factoring n since the prime factorization of n can be used to find e and d, and vice versa. This problem of factoring, which is thought to be a hard problem in itself, adds to the security of the cryptosystem. RSA can be considered the composite analog of the Pohlig-Hellman Cipher.

1.3. **Diffie-Hellman Key Agreement.** If Alice and Bob want to communicate with each other secretly, then, in a private key system, they need to agree on a key in which to encrypt their messages. They first choose a public g and p. Suppose Alice chooses her secret key to be α and Bob chooses his to be β. Then Alice and Bob exchange the following:

$$a \equiv g^{\alpha} \pmod{p} \longleftrightarrow b \equiv g^{\beta} \pmod{p}$$

After receiving these messages, Alice and Bob exponentiate with respect to their secret keys again, thus resulting in both parties having

$$g^{\alpha\beta} \pmod{p}$$

because of the commutativity of multiplication in the integers. This value will be the secret key Alice and Bob use to encrypt messages back and forth between each other. Even if an eavesdropper, Eve, were to intercept $a,\ b,\ g,$ and p, it is thought to be difficult to find $g^{\alpha\beta}$ because

of what is known as the Diffie-Hellman Problem. It can be shown that if the DHP can be solved, then so can the DLP, but not vice versa.

Although the discrete logarithm problem is a hard problem, we will try to exploit it by studying the structure of the discrete exponentiation function given in Equation (1) through a discrete exponentiation functional graph. A *functional graph* is a directed graph with an associated function, $f(x)$. In our particular case, the function is Equation (1), making it a discrete exponentiation functional graph. The *nodes* of the graph represent the elements of the function's domain and codomain. For each x, there is a directed edge (an arrow) from the node representing x to the node representing $f(x)$. The *out-degree* of a node a is the number of elements b such that $f(a) = b$. Since we study only functional graphs, the out-degree of each node is exactly 1. The *in-degree* of a node b is defined as the number of elements a such that $b = f(a)$. A node with an in-degree greater than 0 is an *image node*, and those with in-degree equal to 0 are *terminal nodes*. If every node of a functional graph has in-degree 0 or m, then we say that the graph is an *m-ary graph*. A few more definitions concerning the structure of a functional graph are needed for the remainder of this paper.

If we let $f : S \to S$ be the transition function, then the edges in the functional graph can be expressed as the ordered pair $\langle x, f(x)\rangle$ for $x, f(x) \in S$. By applying the pigeonhole principle and noting that the cardinality of S is finite, say n, we can say that by starting at any arbitrary point u_0 and following the sequence $u_1 = f(u_0), u_2 = f(u_1), ...,$ there must be a $u_i = u_j$ after at most n iterations. Suppose u_i occurs before u_j in the sequence of nodes. In this case, the *tail length* is the number of iterations from u_0 to u_i, and the *tail nodes* are those nodes from u_0 to u_{i-1}. The *cycle length* is the number of iterations from u_i to u_j, and the *cyclic nodes* are the nodes from u_i to u_{j-1}. In more natural

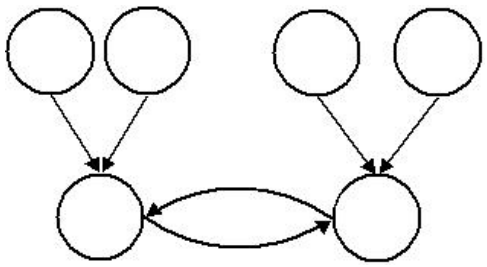

FIGURE 1. Example Ternary Graph. The image nodes are those with arrowheads attached, and tail nodes are those with only arrowtails protruding.

graphical terms, the cycle length is the number of edges (or equivalently nodes) involved in the directed path from u_i to itself, whereas the tail length is the number of edges from u_0 to u_i. A *component* is the maximal set of connected nodes. Since each node has an out-degree of exactly one, each cycle with the trees grafted onto its nodes will form a connected component.

The question becomes, if you were to choose, from all functional graphs, a sample of discrete exponentiation functional graphs and a random sample of arbitrary functional graphs, what similarities would there be? If DEFGs share many characteristics with random functional graphs of the same indegrees, and there are no discernible differences, then it would be safe to assume that DEFGs behave as if a random function were being imposed. This implies that a hacker like Eve would be unable to use any sort of attack that guesses at the structure of the functional graph, because there would be no defining characteristics on which to rely.

For our study, we will consider the composite case where $n = p^b$ for some odd prime p and integer $b > 1$. The reason for this is because this value of n contains a primitive root, necessary for the general encryption methods we have encountered.[1] A few encryption methods using the discrete logarithm problem with a composite modulus include Murakami and Kasahara's ID-based scheme [5], Girault's ID-based scheme [4], and Pointcheval's identity authentication system [7].

We will start by looking at previous work using functional graphs to try exploiting the discrete logarithm problem, then observe what we can about the composite case before comparing DEFGs with composite moduli to random functional graphs.

2. Previous Work

2.1. **Dan Cloutier 2005.** In 2005, Dan Cloutier began work on his senior thesis [3] under the guidance of Dr. Joshua Holden. He looked at permutations (1-ary or unary functional graphs) and binary (2-ary) functional graphs using the function found in (1) with a prime modulus. He developed generating functions, combinatorial objects, to find theoretical statistics for the aspects of a random graph structure produced by the discrete exponentiating function. The structure characteristics looked at by Cloutier were average number of components, average number of cyclic nodes, average number of image nodes, average cycle

[1] In [5], Murakami and Kasahara use a general composite modulus, but instead of a primitive root, they use what they call the *maximal generating element*. In our case, the primitive root is the maximal generating element.

length as seen from a node, average tail length as seen from a node, and the maximum cycle length. After writing a C++ program to calculate all possible unary and binary graphs produced from a given prime modulus p, he was able to compare the statistics from the generated graphs and the random graphs. What he found was, if n was prime, then the structure of the discrete exponentiation functional graph was very similar to that of a random unary or binary graph.

2.2. **Max Brugger and Christina Frederick 2007.** In 2007, Max Brugger and Christina Frederick [2] continued Cloutier's work, but on ternary (3-ary) functional graphs with a prime modulus p. The aspects they studied were average number of components, average number of cyclic nodes, and average cycle length as seen from a node. In his senior thesis, Brugger [1] looked at average tail length as seen from a node in addition to those from his work with Frederick. They developed more complex generating functions to find statistics for a random graph of size $p-1$, since 0 and p are not included in the mapping. The equations for these generating functions, plus the one Brugger developed, can be found in Theorem 2.1.

Theorem 2.1. *The exponential generating functions for the total number of components, total number of cyclic nodes, total cycle length as seen from a node, and total tail length as seen from a node, in a random ternary functional graph of size n are*

$$(2)\quad \textit{Number of Components} = \left[\frac{d}{du}e^{uc(x)}\right]_{u=1} = \frac{1}{2}x^3 + \frac{13}{24}x^6 + \frac{83}{144}x^9 + \frac{355}{576}x^{12} + O(x^{15})$$

$$(3)\quad \textit{Number of Cyclic Nodes} = \left[\frac{d}{du}e^{\ln\frac{1}{1-\frac{1}{2}uxt(x)^2}}\right]_{u=1} = \frac{1}{2}x^3 + \frac{2}{3}x^6 + \frac{29}{36}x^9 + \frac{17}{18}x^{12} + O(x^{15})$$

$$(4)\quad \textit{Cycle Length} = \left[\frac{d^2}{dudw}e^{c(x)}\ln\frac{1}{1-\frac{1}{2}uwxt(wx)^2}\right]_{u=1,w=1} = \frac{3}{2}x^3 + \frac{13}{4}x^6 + \frac{43}{8}x^9 + \frac{191}{24}x^{12} + O(x^{15})$$

$$\text{(5)}\quad \textit{Tail Length} \qquad = \qquad \left[\frac{d}{du}\frac{xut(x)}{(1-\frac{1}{2}xt(x)^2)^2(1-\frac{1}{2}uxt(x)^2)}\right]_{u=1}$$

where

$$t(x)=\frac{xt(x)^3}{3!}+x \quad \text{and} \quad c(x)=\ln\left(\frac{1}{1-\frac{1}{2}xt(x)^2}\right).$$

From [2], we know that the total number of graphs for ternary functional graphs can be found from $g(x)=e^{c(x)}$, where $c(x)$ comes from Theorem 2.1. Brugger and Frederick then show that to find the averages, you must use a "normalizer." Thus the equations for the desired statistics are

$$\text{average number of components} = \frac{\text{total number of components}}{\text{total number of graphs}}$$

$$\text{average number of cyclic nodes} = \frac{\text{total number of cyclic nodes}}{\text{total number of graphs}}$$

$$\text{average cycle length} = \frac{\text{total cycle length}}{\text{number of nodes} \cdot \text{total number of graphs}}$$

$$\text{average tail length} = \frac{\text{total tail length}}{\text{number of nodes} \cdot \text{total number of graphs}}$$

Using Cloutier's code, Brugger and Frederick also found that the structure of the ternary graphs produced from Equation (1) was much like that of a random ternary graph.

3. Composite Moduli Case

We now have to decide how to map this function given by Equation (1) with a modulus of $n=p^b$. If we follow what Cloutier and Brugger and Frederick did, we would map

$$\{1,2,3,\ldots,n-1\} \longrightarrow U_n$$
$$x \longmapsto g^x \pmod{n}$$

Unfortunately, this does not produce an m-ary mapping as we hoped for, as seen in Figure 2. Without an m-ary mapping, the work by

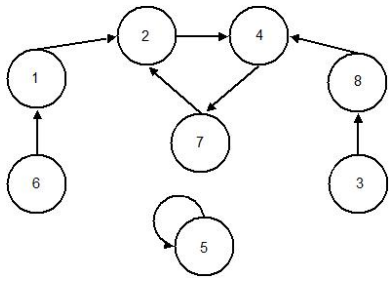

FIGURE 2. $x \longmapsto 2^x \pmod 9$

Brugger and Frederick is of no use to us. However, if we consider the mapping of

$$\{1, 2, 3, \ldots, \operatorname{lcm}(n, \phi(n))\} \longrightarrow U_n \tag{6}$$
$$x \longmapsto g^x \pmod n$$

we can force the mapping to be m-ary. In fact, Theorem (3.1) shows that we can actually force it to be a p-ary mapping.

Theorem 3.1. *Consider the mapping*

$$\{1, 2, 3, \ldots, \operatorname{lcm}(n, \phi(n))\} \longrightarrow U_n$$
$$x \longmapsto g^x \pmod n.$$

If g is a primitive root modulo $n = p^b$ for an odd prime p, then the resulting graph is p-ary.

Proof. Let $\operatorname{lcm}(n, \phi(n)) = m\phi(n)$. Note that since g is a primitive root, it has order $\phi(n)$, so the first $\phi(n)$ domain elements are mapped to distinct elements. To see that if x maps to a certain element, say y, then $x + k\phi(n)$, for some positive integer k, maps to y as well, note that by Euler's Theorem, if $\gcd(x, n) = 1$, then $x^{\phi(n)} \equiv 1 \pmod n$. Thus,

$$g^{x+k\phi(n)} = g^x g^{k\phi(n)} \equiv g^x (g^{\phi(n)})^k \equiv g^x \pmod n$$

Say that the first $\phi(n)$ elements, denoted by x_1, x_2, ..., $x_{\phi(n)}$, map to y_1, y_2, ..., $y_{\phi(n)}$, respectively. Then the following shows the rest of the mapping.

$$\begin{array}{llll}
x_1 \mapsto y_1 & x_1 + \phi(n) \mapsto y_1 & \cdots & x_1 + (m-1)\phi(n) \mapsto y_1 \\
x_2 \mapsto y_2 & x_2 + \phi(n) \mapsto y_2 & \cdots & x_2 + (m-1)\phi(n) \mapsto y_2 \\
\vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \cdots & \vdots \\
x_{\phi(n)} \mapsto y_{\phi(n)} & x_{\phi(n)} + \phi(n) \mapsto y_{\phi(n)} & \cdots & x_{\phi(n)} + (m-1)\phi(n) \mapsto y_{\phi(n)}
\end{array}$$

One can easily verify that all $m\phi(n)$ domain elements are accounted for. Thus, each y_i has m input values, meaning the graph is m-ary. To see that $m = p$, we know that

$$m = \frac{\text{lcm}(n, \phi(n))}{\phi(n)}$$

or, equivalently

$$m = \frac{\text{lcm}(p^b, \phi(p^b))}{\phi(p^b)}.$$

Then,

$$m = \frac{\text{lcm}(p^b, \phi(p^b))}{\phi(p^b)} = \frac{\frac{p^b \cdot \phi(p^b)}{\gcd(p^b, \phi(p^b))}}{\phi(p^b)}$$
$$= \frac{p^b}{\gcd(p^b, p^{b-1}(p-1))} = \frac{p^b}{p^{b-1}} = p.$$

□

Taking the example of mapping $x \longmapsto 2^x \pmod 9$ from Figure 2, we can now transform it into a ternary graph as shown in Figure 3. We will refer to the mapping given by Equation (6) as the LCM method for the rest of the paper.

Using the LCM method, we can also find the number of m-ary functional graphs all possible generators g provide. The values of g and the modulus n will determine the general structure of the graph. The relationship between these numbers given in Theorem 3.2 will give us the in-degree of each node.

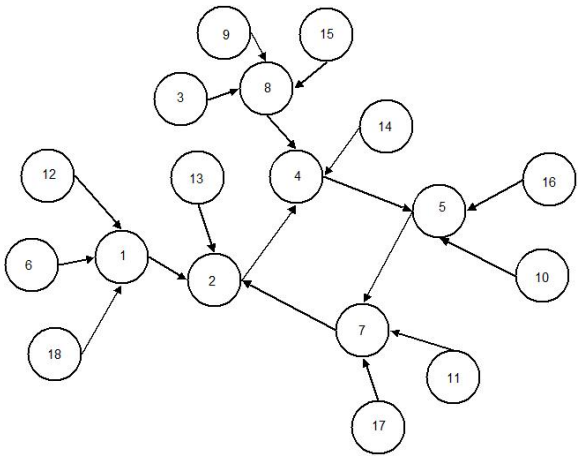

FIGURE 3. $x \longmapsto 2^x \pmod 9$, now a ternary graph

Theorem 3.2. *Let n be a composite integer of the form $n = p^b$ or $n = 2p^b$ for an odd prime p and integer $b > 1$. Let m be any positive integer that divides $\phi(n)$ and let $\operatorname{lcm}(n, \phi(n)) = k\phi(n)$. Then there are $\phi(\frac{\phi(n)}{m})$ km-ary functional graphs produced by the mappings*

$$x \mapsto g^x \pmod{n}$$

for g, between 1 and n, and n. Furthermore, if r is any primitive root of n, and $g \equiv r^a \pmod{n}$, then the values of g that produce a km-ary graph are precisely those for which $\gcd(a, \phi(n)) = m$.

Proof. Let r be a primitive root of n. Choose a and t so that $g \equiv r^a \pmod{n}$ and $y \equiv r^t \pmod{n}$ for g and y in Equation (1). In terms of r this becomes

$$r^{ax} \equiv r^t \pmod{n}$$

Since r is a primitive root, the solutions for this equation are the same as for

$$ax \equiv t \pmod{\phi(n)}$$

Let $m = \gcd(a, \phi(n))$. By Theorem 2.17 in [6], there are m solutions between 1 and $\phi(n)$ if $m|t$, and no solutions if $m \nmid t$. Thus there will be km solutions between 1 and $k\phi(n) = \operatorname{lcm}(n, \phi(n))$. Since each solution is one directed edge to y, the in-degree of y is either km or 0, depending on whether $m|t$. The values of a which satisfy $m = \gcd(a, \phi(n))$ are precisely the same as those which satisfy

$$1 = \gcd\Big(\frac{a}{m}, \frac{\phi(n)}{m}\Big).$$

There are $\phi(\frac{\phi(n)}{m})$ values of $\frac{a}{m}$ less than $\frac{\phi(n)}{m}$ and relatively prime to $\frac{\phi(n)}{m}$. Thus there are $\phi(\frac{\phi(n)}{m})$ km-ary functional graphs. □

Since we do not have any generating functions with in-degree greater than 3, we look for $km = 3$ because we are only able to compare ternary graph structures. This provides only two cases because k and m are both integers.

i) $k = 1,\ m = 3$
If this is true, then $\operatorname{lcm}(n, \phi(n)) = \phi(n)$, which implies that $n|\phi(n)$. This cannot be true by the definition of ϕ unless $n = \phi(n)$. But this implies that $n = 1$, which is not an option based on our definition of $n = p^b$. The following must be true then.

ii) $k = 3,\ m = 1$
This case implies that $\operatorname{lcm}(n, \phi(n)) = 3\phi(n)$. By some elementary

number theory we find that

$$3\phi(n) = 3\phi(p^b) = \operatorname{lcm}(p^b, \phi(p^b)) = \frac{p^b\phi(p^b)}{\gcd(p^b, \phi(p^b))}$$
$$= \frac{p^b\phi(p^b)}{p^{b-1}} = p\phi(p^b) \Rightarrow 3 = p.$$

Thus our prime p must be 3. This is also useful because now $\frac{\phi(n)}{m}$ is guaranteed to be an integer since $m = 1$, giving us a nonzero number of ternary graphs.

There is one major problem with the LCM method. It is that, even though we can form a p-ary graph, this does us no good unless we can find generating functions for p-ary graphs where $p > 3$. Restricting ourselves to one value of p will not allow us to fully explore the structure of the functional maps produced with composite moduli. If we find that developing these generating functions is too difficult, then we will need to find another way of comparing our functional graphs to random graphs.

4. Statistics

Having shown that we must choose powers of 3 in order to generate ternary graphs using the LCM method, we next have to choose which powers of 3 we will use to compare the theoretical findings against the observed results. Using Maple, I generated the powers of 3 up to 3^{50}. If the $\operatorname{lcm}(3^b, \phi(3^b))$ was greater than 1000, but less than 150,000, then that power of 3 was included in the statistics that follow. The lower bound is based on the fact that the C++ code used for the observed values did not operate properly for values less than 1000. The upper bound was obtained due to the computation time needed by Maple to calculate the coefficient of the generating functions for a given value. Numbers greater than 100,000 required nearly 5 hours to evaluate one statistic. There were 5 values for 3^b in which the $\operatorname{lcm}(3^b, \phi(3^b))$ was greater than 1,000 and less than 100,000: 3^6, 3^7, 3^8, 3^9, and 3^{10}.

We will compare the structures of the graphs using average number of terminal nodes, average number of components, average number of cyclic nodes, average cycle length as seen from a node, and average tail length as seen from a node.

4.1. Theoretical Results.

Using the same generating functions used by Max Brugger in [1] and discussed in Section 2, I was able to collect the theoretical values for the five statistics. I used Maple to transform

the exponential generating functions into differential equations. Then I converted the differential equations to recursive equations that were used to calculate the coefficients for the ternary functional graphs associated with our values of the $\text{lcm}(3^b, \phi(3^b))$. These theoretical statistics give us the expected number of whichever aspect of the graph structure we are interested in.

4.2. **Observed Results.** In order to obtain the observed results, we used a version of Daniel Cloutier's C++ code, modified by Nathan Lindle and later by Andrew Hoffman. This version of the code worked only with prime moduli, but few changes were needed to make it compatible with composites of the form $n = p^b$. The only major change was editing the search for a primitive root function. The C++ performed the same as in Cloutier's and Brugger/Frederick's cases by finding the smallest primitive root of a given modulus n and then generating all primitive roots from that primitive root. Using these primitive roots, all possible m-ary graphs are found in order to collect the statistics. The program would keep track of all the needed statistics, then report them back to the user to compare against the theoretical data.

4.3. **Analysis of Data.** Tables 1, 2, 3, 4, and 5 show the comparisons between the theoretical values obtained from the ternary generating functions and the observed values from the modified C++ program. The average number of terminal nodes came out to be the exact same as expected, which is due to the fact that the value of the number of terminal nodes can actually be proven. Cloutier proves the following for the prime case in [3], but it can easily be modified to our composite case.

Theorem 4.1. *Let $c = \text{lcm}(n, \phi(n))$, the number of nodes of the graph. The number of image nodes in an m-ary graph is $\frac{c}{m}$. Equivalently, the number of terminal nodes is $c - \frac{c}{m}$.*

In our case, m is 3, meaning two-thirds of our nodes should be terminal nodes. This claim is backed up by our results. In all of the other statistics, the observed and theoretical differ by quite a bit. The average number of components is greater from our C++ program than the expected value given by the generating functions by almost double. However, the average cycle length as seen from a node and the average tail length as seen from a node are both less than the expected values, considerably different from what the generating functions predicted. The average number of cyclic nodes begins higher than expected, but around $n = 3^8$, it lags behind the theoretical value.

TABLE 1. Average Number of Terminal Nodes

n	$lcm(n, \phi(n))$	No. of Ternary Graphs	Theoretical Values	Observed Values	Rel. Error
3^6	1458	162	972	972	0.000
3^7	4374	486	2916	2916	0.000
3^8	13122	1458	8748	8748	0.000
3^9	39366	4374	26244	26244	0.000
3^{10}	118098	13122	78732	78732	0.000

TABLE 2. Average Number of Components

n	$lcm(n, \phi(n))$	No. of Ternary Graphs	Theoretical Values	Observed Values	Rel. Error
3^6	1458	162	3.93873	8.77778	1.22858
3^7	4374	486	4.48378	9.65432	1.15317
3^8	13122	1458	5.03221	10.64198	1.11477
3^9	39366	4374	5.58042	11.77229	1.10957
3^{10}	118098	13122	6.12910	12.73891	1.07843

TABLE 3. Average Number of Cyclic Nodes

n	$lcm(n, \phi(n))$	No. of Ternary Graphs	Theoretical Values	Observed Values	Rel. Error
3^6	1458	162	33.17953	52.59259	0.58509
3^7	4374	486	57.94893	74.62963	0.28785
3^8	13122	1458	100.85403	101.74897	0.00887
3^9	39366	4374	175.16974	147.48423	0.15805
3^{10}	118098	13122	303.88942	207.83905	0.31607

TABLE 4. Average Cycle Length (as seen from a node)

n	$lcm(n, \phi(n))$	No. of Ternary Graphs	Theoretical Values	Observed Values	Rel. Error
3^6	1458	162	17.08977	5.07654	0.70295
3^7	4374	486	29.47446	7.28542	0.75282
3^8	13122	1458	50.92701	9.40146	0.81539
3^9	39366	4374	88.08487	13.19326	0.85022
3^{10}	118098	13122	152.44471	18.31171	0.87987

Although each statistic by itself is worrying, the fact that the average number of components is higher than expected, but the average cycle length as seen from a node and the average tail length as seen from a node are both less than the expected values makes sense. If there are

TABLE 5. Average Tail Length (as seen from a node)

n	$lcm(n, \phi(n))$	No. of Ternary Graphs	Theoretical Values	Observed Values	Rel. Error
3^6	1458	162	11.39318	5.30983	0.53395
3^7	4374	486	19.64964	7.61851	0.61228
3^8	13122	1458	33.95134	10.97598	0.67671
3^9	39366	4374	58.82325	14.92264	0.74631
3^{10}	118098	13122	101.62980	20.84103	0.79493

more components that the nodes must be spread throughout, it is not unreasonable that the average cycle and tail length would both be less since there are fewer nodes in each component to form a cycle or a tail.

5. Conclusion

While our statistics leave us in a large predicament of what to make of our results, there are at least three possible explanations for our findings. One is that the C++ code in which the observed results were obtained has a programming bug. This is possible, but not likely given that our terminal nodes came out exactly the way they should have. Second is that we have found some evidence that the discrete exponentiation graph is unlike a random map when the modulus is a composite of the form $n = p^b$. This would be a starting point to finding a way to exploit the discrete logarithm problem. The last possibility is that the theoretical generating functions we are using are wrong. Since we are mapping (1) in an unconventional manner, it is possible that the generating functions used for the prime case are not applicable to our case. If this is the case, new generating functions must be developed to account for the changes.

Composite cases to consider for future work are $n = 2p^b$ for an odd prime p and integer $b > 1$, and $n = pq$ for odd primes p and q. The case where $n = 2p^b$ is the only other composite form which contains a primitive root. The only question is to see if an m-ary mapping can be naturally generated, and if not, be forced into one using a method similar to the LCM method. The $n = pq$ case looks at moduli in the form of the RSA numbers, used in the RSA Cryptosystem. These numbers do not have primitive roots, so it must be determined how to work with them in order to obtain an m-ary graph, or to use an entirely new approach. A possible starting point would be to look at the maximal generating element described in [5].

Another area of future work would be to work on developing a

method for finding the associated generating function for a given m-ary graph for $m > 3$. As the in-degree for these generating functions increases, the more complex they become, making it nearly infeasible to be derived by hand. In order to see if the LCM method is a plausible agent in helping exploit the structure of these discrete exponentiation graphs, it needs to be tested for primes other than 3. Developing these higher order generating functions would help in verifying the results given by the LCM method.

References

[1] Brugger, Max. Exploring the Discrete Logarithm with Random Ternary Graphs. Senior thesis, Oregon State University, 2008.

[2] Brugger, Max and Christina Frederick. The Discrete Logarithm Problem and Ternary Functional Graphs. *Rose-Hulman Institute of Technology Undergraduate Mathematics Journal* **8** (2), 2007.

[3] Cloutier, Dan. Mapping the discrete logarithm. Undergraduate thesis, Rose-Hulman Institute of Technology, 2005.

[4] Girault, Marc. An identity-based identification scheme based on discrete logarithms modulo a composite number. *Lecture Notes in Computer Science* **73**, 1991. 481-486.

[5] Kasahara, Masao, and Yasuyuki Murakami. A Discrete Logarithm Problem over Composite Modulus. *Electronics and Comm. in Japan* **76** (12), 1993. 37-46.

[6] Niven, Ivan, Herbert S. Zuckerman, Hugh L. Montgomery. *An Introduction to the Theory of Numbers.* John Wiley and Sons, Inc, 1991.

[7] Pointcheval, David. The Composite Discrete Logarithm and Secure Authentication. *Lecture Notes in Computer Science* **1751**, 2000. 113-128.

[8] Rosen, Kenneth H. *Elementary Number Theory and its Applications, 3rd Edition.* Addison Wesley, 1993.

E-mail address: mlm05h@acu.edu