2009

Analysis of a man-in-the-middle attack on the Diffie-Hellman key exchange protocol

Geary, Aaron C.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ANALYSIS OF A MAN-IN-THE-MIDDLE ATTACK ON THE DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL

by

Aaron C. Geary

September 2009

Thesis Co-Advisors: Pantelimon Stanica
Valery Kanevsky

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2009	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Analysis of a Man-in-the-Middle Attack on the Diffie-Hellman Key Exchange Protocol	5. FUNDING NUMBERS
6. AUTHOR(S) Aaron C. Geary	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

The ability to distribute cryptographic keys securely has been a challenge for centuries. The Diffie-Hellman key exchange protocol was the first practical solution to the key exchange dilemma. The Diffie-Hellman protocol allows two parties to exchange a secret key over unsecured communication channels without meeting in advance. The secret key can then be used in a symmetric encryption application, and the two parties can communicate securely. However, if the key exchange takes place in certain mathematical environments, the exchange becomes vulnerable to a specific man-in-the-middle attack, first observed by Vanstone [1]. We explore this man-in-the-middle attack, analyze countermeasures against the attack, and extend the attack to the multi-party setting.

14. SUBJECT TERMS Cryptography, Diffie-Hellman, Man-in-the-Middle Attack	15. NUMBER OF PAGES 75
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ANALYSIS OF A MAN-IN-THE-MIDDLE ATTACK ON THE DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL

Aaron C. Geary
Lieutenant, United States Navy
B.S., United States Naval Academy, 2003

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

and

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Aaron C. Geary

Approved by: Pantelimon Stanica
Thesis Co Advisor

Valery Kanevsky
Thesis Co Advisor

Carlos Borges
Chairman, Department of Applied Mathematics

Dan Boger
Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to distribute cryptographic keys securely has been a challenge for centuries. The Diffie-Hellman key exchange protocol was the first practical solution to the key exchange dilemma. The Diffie-Hellman protocol allows two parties to exchange a secret key over unsecured communication channels without meeting in advance. The secret key can then be used in a symmetric encryption application, and the two parties can communicate securely. However, if the key exchange takes place in certain mathematical environments, the exchange becomes vulnerable to a specific man-in-the-middle attack, first observed by Vanstone [1]. We explore this man-in-the-middle attack, analyze countermeasures against the attack, and extend the attack to the multi-party setting.

THIS PAGE INTENTIONALLY LEFT BLANK

DISCLAIMER

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

Heartfelt thanks to both of my advisors, Professor P. Stanica of the Math Department and Professor V. Kanevsky of the Information Sciences Department. I appreciate their in-depth reading and comments. I also thank Mr. Bard Mansager, without whom I never would have considered a dual-degree. A special thanks to Professor Stanica for helping me arrive at a thesis topic, his continued sharing of expertise in all matters of mathematics, and his never ending patience with the many ideas I ran by his desk. I could not have written this without his guidance.

Thanks to my amazing wife, Sandy. Without her love and support, I would be lost in this life. Lastly, I thank all my friends and family. Without them, all else is meaningless, even secure exchange of cryptographic keys!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The ability to communicate securely has been a challenge for millennia. For as long as people have tried to exchange private information, others have tried to compromise their privacy. In the modern communications environment, radio frequency communications and worldwide digital networks, such as the Internet, compound the problem. Both are susceptible to eavesdropping, often times trivially. By simply placing an antenna in the region of a radio frequency broadcast or tapping a wire anywhere between two nodes on a digital network, an uninvited third party can easily gain access to seemingly private correspondence. The field of cryptography—the practice and study of hiding information—has made enormous progress combating the eavesdropping threat.

Cryptography and encryption/decryption methods fall into two broad categories: symmetric and public key. In symmetric cryptography, sometimes called classical cryptography, parties share the same encryption/decryption key. Therefore, before using a symmetric cryptography system, the users must somehow come to an agreement on a key to use. An obvious problem arises when the parties are separated by large distances, which is commonplace in today's worldwide digital communications. If the parties did not meet prior to their separation, how do they agree on the common key to use in their crypto system without a secure channel? They could send a trusted courier to exchange keys, but that is not feasible, if time is a critical factor in their communication.

The problem of securely distributing keys used in symmetric ciphers has challenged cryptographers for hundreds of years. If an unauthorized user gains access to the key, the cryptographic communication must be considered broken. Amazingly, in 1977, Whitfield Diffie and Martin Hellman published a paper in which they presented a key exchange protocol that provided the first practical solution to this dilemma. The protocol, named the Diffie-Hellman key exchange (or key agreement) protocol in their honor, allows two parties to derive a common

secret key by communications over an unsecured channel, while sharing no secret keying material a priori [2]. While Diffie and Hellman have received recognition for creating the protocol, it later emerged that the Government Communications Headquarters (GCHQ), a British intelligence agency, had independently invented a similar protocol a few years before Diffie and Hellman published their breakthrough paper. However, the British government classified their findings and the results were not released to the public until 1997 [3].

The Diffie-Hellman protocol relies on the difficulty of solving discrete logarithms in finite fields and the related intractability of the Diffie-Hellman problem. Due to the difficulty of solving these mathematical problems, an eavesdropper is unable to compute efficiently the secret key with any or all of the information intercepted in the open communication channel. Once the secret key has been exchanged successfully between the two parties, they may proceed by using the key in their symmetric crypto system.

Before conducting the key exchange using the Diffie-Hellman protocol, the parties must agree on a prime number that defines the mathematical environment in which the key exchange will take place. If the prime number is large enough, a brute force attack to find the secret key becomes infeasible. However, if the two parties agree on certain prime numbers, an active adversary can compromise their communication. Using number theory, a man-in-the-middle attack becomes possible if the prime number that defines the environment can be broken down into the form of $p = Rq + 1$, where R is a “small” integer and q is a “large” prime. If possible, the attacker can then modify the messages between the two parties so that they will both derive a key that belongs to a subgroup of size R. If R is small enough, the attacker can search the keyspace in a reasonable amount of time, determine the key the parties agreed to, and eavesdrop on their communication.

This thesis investigates the Diffie-Hellman protocol and the difficulty of the discrete logarithm problem the protocol relies on. We then analyze the man-in-middle attack described above by developing an algorithm to conduct the attack, estimate the complexity involved in executing the attack, and approximate the amount of prime numbers that are vulnerable. We then consider several proposed methods to defend against the attack and demonstrate their effectiveness. Finally, we extend the attack to several multi-party variants of the protocol and demonstrate their potential vulnerability.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND REVIEW

Before beginning a discussion of the Diffie-Hellman protocol and the man-in-the-middle attack, we investigate and present some basic definitions and theorems. This information is available in any standard algebra text, such as Fraleigh's *Abstract Algebra* [4], or discrete mathematics text, such as Rosen's *Discrete Mathematics and Its Applications* [5]. It is assumed the reader is familiar with common mathematical, logical, and set notation.

We conclude the chapter with a brief discussion of computational complexity and primality testing, which will be useful in our analysis of the man-in-the-middle attack.

A. NUMBER THEORY

If a and b are integers and $a \neq 0$, we say that a *divides* b if there is an integer c such that $b = ac$. When a divides b we say that a is a *factor* of b and that b is a *multiple* of a. The notation $a|b$ denotes a divides b. Given two integers a and b, both non-zero, the largest integer d such that $d|a$ and $d|b$ is called the *greatest common divisor* of a and b. The greatest common divisor of a and b is denoted by $\gcd(a,b)$. The integers a and b are *relatively prime,* if their greatest common divisor is one.

Every positive integer greater than one is divisible by at least two integers, itself and one. If these are its only factors, we call this integer *prime*. A positive integer that is greater than one, and not prime, is called *composite*. The primes are the building blocks of positive integers. The Fundamental Theorem of Arithmetic states that every positive integer greater than one can be written uniquely as a product of two of more primes, where the prime factors are written in order of nondecreasing size. Given a positive integer, n, let the prime factorization of n be denoted by

$$n = \prod_{i=1}^{k} p_i^{\alpha_i}$$

In some situations, we care only about the remainder of an integer when it is divided by some specified positive integer, denoted by m. If a and b are integers, then a is *congruent to* b *modulo* m if m divides $a-b$. We use the notation $a \equiv b \pmod{m}$ to indicate that a is congruent to b modulo m. Note that $a \equiv b \pmod{m}$ if and only if $a \pmod{m} = b \pmod{m}$. Also, if n divides a then a is congruent to zero modulo n.

The great French mathematician Pierre de Fermat (1601–1655) demonstrated that the congruence

$$a^{p-1} \equiv 1 \pmod{p}$$

holds when p is a prime, and this gives us a theorem that will prove crucial in our analysis of the man-in-the-middle attack.

Fermat's Theorem [4]**:** *If* $a \in \mathbb{Z}$ *and* p *is a prime not dividing* a*, then* p *divides* $a^{p-1}-1$*, that is,* $a^{p-1} \equiv 1 \pmod{p}$*.*

Euler gave a generalization of Fermat's theorem, but we must first define ***Euler's Totient Function***. Commonly referred to as ***Euler's Phi Function***, the function gives the number of integers less than or equal to n which are relatively prime to n, and is denoted by $\phi(n)$. It is not hard to show that, if $n = \prod_{i=1}^{k} p_i^{\alpha_i}$, then

$$\phi(n) = n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

Euler's Theorem [4]**:** *If* $a \in \mathbb{Z}$ *and is relatively prime to* n*, then* $a^{\phi(n)}-1$ *is divisible by* n*, that is,* $a^{\phi(n)} \equiv 1 \pmod{n}$*.*

In several cases, this thesis will involve systems of linear congruences. The ***Chinese Remainder Theorem [CRT]***, named after the Chinese heritage of problems involving systems of linear congruences, states that when the moduli of

a system of linear congruences are pairwise relatively prime, there is a unique solution of the system modulo the product of the moduli.

[*CRT*] [5]**:** *Let* $m_1, m_2, ..., m_n$ *be pairwise relatively prime positive integers and* $a_1, a_2, ..., a_n$ *arbitrary integers. Then the system*

$$\begin{cases} x \equiv a_1 (\mathrm{mod}\, m_1), \\ x \equiv a_2 (\mathrm{mod}\, m_2), \\ . \\ . \\ . \\ x \equiv a_n (\mathrm{mod}\, m_n) \end{cases}$$

has a unique solution modulo $m = m_1 m_2 ... m_n$*. (That is, there is a solution* x *with* $0 \le x < m$*, and all other solutions are congruent modulo* m *to this solution.)*

B. GROUP THEORY

A ***group*** $\langle G, * \rangle$ is a set G, closed under a binary operation $*$, such that the following axioms are satisfied:

Associativity: For all $a, b, c \in G$, $(a * b) * c = a * (b * c)$

Identity: There is an element e in G such that for all $x \in G$, $e * x = x * e = x$.

Inverse: Corresponding to each $a \in G$, there is an element a' in G such that $a * a' = a' * a = e$.

A group that also satisfies the commutative property is referred to as an ***abelian*** (or ***commutative***) group.

Commutativity: For all $a, b \in G$, $a * b = b * a$.

A group G is said to be a ***finite group,*** if the set G has a finite number of elements. In this case, the number of elements is called the ***order*** of G, denoted by $|G|$. This thesis is interested only in finite groups.

If a subset H of a group G is closed under the binary operation of G and if H with the induced operation from G is itself a group, then H is a subgroup of G. We shall let $H \leq G$ or $G \geq H$ mean that H is a subgroup of G, and $H < G$ or $G > H$ shall mean $H \leq G$ but $H \neq G$.

An example of a group is the set of congruence classes of the integers modulo n. Given a positive integer n, we denote a congruence class by $|a|_n$ which is the set of all integers congruent to a modulo n. The set of congruence classes of n is denoted by

$$\mathbb{Z}_n = \{[0]_n, [1]_n, ..., [n-2]_n, [n-1]_n\}$$

This set forms a group under addition where $[a]_n + [b]_n = [a+b]_n$ and is denoted by $\langle \mathbb{Z}_n, + \rangle$. We can easily inspect a group using a group table. Table 1 is a group table for $\mathbb{Z}_5$ under addition. The elements of $\mathbb{Z}_5$ are the column and row headings,, with the binary operation (addition in this case), in the upper left corner.

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Table 1. Group Table for $\langle \mathbb{Z}_5, + \rangle$

If n is a prime p, then the set $\mathbb{Z}_p^* = \mathbb{Z}_p - \{[0]_p\}$ forms a group under multiplication modulo n. It is a necessary requirement to remove the zero class because zero has no inverse under multiplication. $\langle \mathbb{Z}_p^*, \cdot \rangle$, the multiplicative group of the set of congruence classes of prime integers, is the structure we will

be focusing on in this thesis. The Diffie-Hellman key exchange protocol sets this group as the environment for the key agreement. If we remove the zero element from the previous example, we have another group table (Table 2), this time with multiplication as the binary operation.

•	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Table 2. Group Table for $\langle \mathbb{Z}_p^*, \cdot \rangle$

Let G be a group and let $a \in G$. Then the subgroup $\{a^n \mid n \in \mathbb{Z}\}$ of G is called the *cyclic subgroup* of G generated by a, and is denoted by $\langle a \rangle$. Further, a *generates* G if $\langle a \rangle = G$. A group G is *cyclic* if there is some element a in G that generates G.

The group $\langle \mathbb{Z}_p^*, \cdot \rangle$ is always cyclic. An important property of cyclic groups is that every subgroup of a cyclic group is also cyclic. Another important property of groups in general is the Theorem of Lagrange.

Lagrange's Theorem [4]: *Let H be a subgroup of a finite group G. Then the order of H is a divisor of the order of G.*

This powerful theorem makes the attack we will analyze later possible. We know the order of $\langle \mathbb{Z}_p^*, \cdot \rangle$ is $p-1$. The two properties mentioned above tell us that any subgroup of $\langle \mathbb{Z}_p^*, \cdot \rangle$ will also be cyclic and the order of the subgroup will be a divisor of $p-1$.

C. FIELD THEORY

A field $\langle F,+,\cdot\rangle$, is a set F together with two binary operations, which we will call addition and multiplication, defined on F such that the following axioms are satisfied:

Addition: $\langle F,+\rangle$ is an abelian group.

Multiplication: $\langle F^*,\cdot\rangle$ is an abelian group.

Distributive: For all $a,b,c \in F$, $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$.

A field F is said to be a ***finite field***, if the set F has a finite number of elements. If F is a finite field, then the multiplicative group is cyclic.

For every prime p and positive integer n, there is exactly one finite field (up to isomorphism) of order p^n. This field $GF(p^n)$ is usually referred to as the ***Galois field*** of order p^n. Oftentimes, the Diffie-Hellman key exchange protocol is described using the environment $GF(p)$ instead of the group $\mathbb{Z}_p^*$. In the group theory section, we described the notion of a generator of a cyclic group. In field theory, specifically in $GF(p)$, the same element that will generate the entire multiplicative group is known as a primitive root. The number of primitive roots of a field $GF(p)$ is $\phi(\phi(p))=\phi(p-1)$.

D. COMPUTATIONAL COMPLEXITY

Before the discussion of primality testing, it is important to understand what makes one test more efficient than another. ***Computational complexity*** involves the study of the efficiency of algorithms based on the time and memory space required to solve a problem of a particular size [5]. Usually, complexities are expressed using the ***Big-O*** notation.

Definition [5]: Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there are constants C and k such that

$$|f(x)| \leq C|g(x)|$$

Whenever $x > k$. [This is read as "$f(x)$ is big-oh of $g(x)$."]

This notation is extremely helpful when comparing algorithms, such as the primality tests we will discuss. We will use the Big-O notation as an upper bound on the amount of operations a test will require. In general, the smaller the upper bound, the more efficient the test is. The more efficient the test is, the quicker it can complete the required steps of an algorithm and give an answer. Thus, using the Big-O notation, we can often quickly decide which test will finish soonest, using fewer resources and less computer time.

The most commonly used functions in Big-O notation are:

$$1, \log n, n, n \log n, n^2, 2^n, n!$$

It is shown that each function in the list is smaller than the succeeding function as n grows without bound [5]. Figure 1 demonstrates this fact.

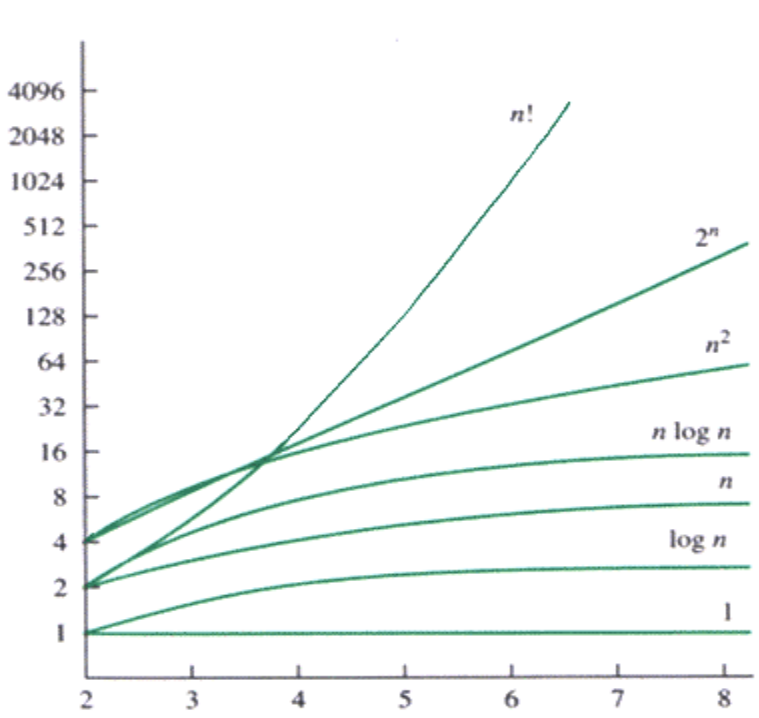

Figure 1. Growth of Functions Used in Big-O Estimates [From 5]

Notice the vertical axis scale is logarithmic, doubling each unit. This causes the exponential function 2^n to appear as a straight line.

An algorithm that is Big-O of a constant has constant complexity. An algorithm that is Big-O of a logarithm has logarithmic complexity, and so on. Table 3 displays the common terminology used to describe the time complexity of algorithm.

Complexity	Terminology
$O(1)$	Constant complexity
$O(\log n)$	Logarithmic complexity
$O(n)$	Linear complexity
$O(n^b)$	Polynomial complexity
$O(b^n)$	Exponential complexity
$O(n!)$	Factorial complexity

Table 3. Computational Complexity Terminology [From 5]

The algorithms we will be concerned with are of polynomial and exponential complexity. The difference between the two can be enormous. Polynomial or better complexities are called ***tractable***, because it is assumed that given a reasonably-sized input, the algorithm will produce an answer in a reasonable amount of time. On the other hand, exponential complexities or worse are called ***intractable***. This is because an extremely large amount of time is usually required to run the algorithm. However, a polynomial complexity algorithm with a very high degree might take longer to run than an exponential complexity algorithm with a small base.

E. PRIMALITY TESTING

We now turn to a topic of critical importance in our analysis of the man-in-the-middle attack. Suppose a large integer is given. How might we quickly be able to tell if the number is prime or composite? Mathematicians have studied this question for millennia, and recently this question has become even more important as modern computing power has granted the ability to test theories on a scale that was at one point inconceivable. A ***primality test*** is an algorithm for determining whether an input number is prime. Primality tests can be divided into two main groups: deterministic and probabilistic. Deterministic primality tests prove with certainty whether a number is prime or composite. Probabilistic primality tests tell us a number is composite or *probably* prime. If a probabilistic

method returns the number is composite, the number is definitely composite. However, if it returns the number as prime, there is a controllably small chance the number is actually composite [6].

Primality testing is currently a topic of great interest and research and is, therefore, very dynamic. We provide descriptions of several deterministic and probabilistic algorithms as background for the reader. It is by no means a comprehensive discussion of every algorithm available. Rather, we use this section as a way to motivate our choice of a primality test for later on when we will need to quickly determine if a given number is prime.

1. Deterministic Primality Tests

a. Trial Division

The simplest primality test is trial division. Trial division is the method of sequentially trying test divisors into a number n so as to partially or completely factor n [6]. We start with the first prime number, 2, and try to divide n by 2. If 2 divides n, we know n is composite and can stop. If 2 does not divide n, we try the next prime number, 3. If 3 divides n, we stop. If not, we try the next prime, and so on. When we reach a trial divisor that is greater than the square root of n, we may stop. If no prime up to the square root of n divides n, then we declare n a prime.

This test is quite computationally intensive. Let $\pi(t)$ be the ***prime counting function***, which counts the number of primes $\leq t$. Trial division requires (in the worst case) about $\pi\left(\sqrt{n}\right) \approx \frac{2\sqrt{n}}{\ln n}$ divisions, if the primes to $\sqrt{n}$ are stored in a database, or even $\frac{\sqrt{n}}{2}$ divisions, if the primes are not stored before the test starts.

b. The n–1 Test

Trial division can be used to test small numbers for primality, but for larger numbers there are better methods [6]. The $n-1$ test is based on Fermat's little theorem, and suggests that we try to factor $n-1$, not n. In 1876, E. Lucas turned Fermat's little theorem into a primality test.

Lucas' Theorem [6]**:** If a, n are integers with $n>1$, and $a^{n-1} \equiv 1 \pmod{n}$, but $a^{(n-1)/q}$ is not congruent to 1, modulo n for every prime $q \mid n-1$, then n is prime.

The most difficult step in implementing the Lucas test is finding the complete factorization of $n-1$. Pocklington strengthened the result by realizing a partial factorization would suffice [6]. In particular, say

$n-1 = FR$, and the complete factorization of F is known. (1)

Pocklington's Theorem**:** Suppose (1) holds and $a^{n-1} \equiv 1 \pmod{n}$ and $\gcd(a^{(n-1)/q} - 1, n) = 1$ for each prime $q \mid F$. Then every prime factor of n is congruent to 1 $\pmod{F}$. (2)

Corollary (n-1 test)**:** If (1) and (2) hold and $F \geq \sqrt{n}$, then n is prime.

Several results have allowed a smaller value of F. These include work done by Brillhart, Lehmer, Selfridge, Konyagin, and Pomerance [6].

The Lucas test and variations of it have a running time of about $O\left((\log n)^3\right)$. The question of finding the "right" base still remains.

c. Elliptic Curve Primality Proving

Elliptic Curve Primality Proving (ECPP) is a class of algorithms that provide certificates of primality using sophisticated results from the theory of elliptic curves. A detailed description of the background, theory, and implementation of the ECPP can be found in Atkin and Morain [7].

ECPP is the fastest known general-purpose primality-testing algorithm. ECPP has a running time of $O\left((\log n)^4\right)$ [7].

d. The AKS Test

In August 2002, the Agrawal-Kayal-Saxena (AKS) primality test was published in a paper titled "Primes is in P" [8]. The result was highly celebrated because of the four properties the test satisfies:

1) It can be used to verify the primality of any given number.
2) The maximum running time is polynomial.
3) The algorithm is deterministic, not probabilistic
4) The algorithm is not conditional on an unproven hypothesis.

There are other algorithms that satisfy three of the four properties, but AKS is the only known test to satisfy all four.

The test is based upon the equivalence

$$(x-a)^n \equiv (x^n - a) \pmod{n}$$

for a coprime to n, which is true if and only if n is prime. This is a generalization of Fermat's Little Theorem and constitutes a primality test by itself. However, the verification of primality would take exponential time, and thus, requires improvement. The AKS test makes use of a related equivalence

$$(x-a)^n \equiv (x^n - a) \pmod{n, x^r - 1}.$$

This equivalence can be checked in polynomial time, with the complexity of the original algorithm being $O\left((\log n)^{12}\right)$. However, recently the complexity has been brought down to $O\left((\log n)^6\right)$ [9].

2. Probabilistic Primality Tests

a. Fermat Primality Test

Based on Fermat's Little Theorem, the Fermat Primality Test is a probabilistic primality test that is the basis for the Miller-Rabin primality test used later on in the thesis.

Recall that by Fermat's Little Theorem, if p is prime and p does not divide a, then $a^{p-1} \equiv 1 \pmod{p}$. If we want to test if a given integer n is prime, we compute $a^{n-1} \pmod{n}$ for several values of a. If the result is not 1 for some value of a, then n is composite. If the result is 1 for many values of a, then we can say that n is *probably* prime.

The reason we can only say *probably* is because the congruence $a^{n-1} \equiv 1 \pmod{n}$ may hold when n is composite. A composite number n is a ***(Fermat) pseudoprime,*** if the congruence $a^{n-1} \equiv 1 \pmod{n}$ holds [6]. Unfortunately, for the Fermat Primality Test, there are infinitely many numbers that the test would call probably prime even if every value of a was computed [6]. These numbers are the so-called ***Carmichael numbers*** and give us reason to look for a test that will only give pseudoprimes for a fixed fraction of the bases attempted. The Miller-Rabin test accomplishes this goal.

b. Miller-Rabin Primality Test

The Miller-Rabin Primality Test is an efficient probabilistic algorithm to test for primality based on the idea of strong pseudoprimes. Consider an odd composite number n and $n-1 = d \cdot 2^s$ with d odd. n is a ***strong pseudoprime*** if either $a^d \equiv 1 \pmod{n}$ or $a^{d \cdot 2^r} \equiv -1 \pmod{n}$ with $r = 0,1,...s-1$. The Carmichael numbers are Fermat pseudoprimes for every base. However, a composite number can only be a strong pseudoprime to at most one quarter of all bases [6].

The algorithm is as follows:

Choose a random integer $a \in [2, n-2]$. If $a^d \neq 1 \pmod n$ and $a^{d \cdot 2^r} \neq -1 \pmod n$ for all $r = 0,1,...s-1$, then a is called a witness and n is composite. Otherwise, n is a strong probable prime to base a.

If $n > 9$ and is odd composite, the probability that the algorithm will fail to produce a witness for n is $< 1/4$. The probability that we fail to find a witness after k iterations is $< 1/4^k$ [6]. We can make this probability as small as we desire with a large number of iterations. For instance, if we wanted to ensure the probability of calling a composite number a prime is less than 10^{-6}, we must compute 10 iterations or more.

As an example, suppose we wanted to determine if the number 341 is prime. First we write $341 - 1 = 340 = 2^2 \cdot 85$. So $s = 2$ and $d = 85$. We randomly select $a = 38$ and proceed with:

$a^d \bmod n = 38^{85} \bmod 341 = 56 \neq 1$

$a^{2^0 d} \bmod n = 38^{85} \bmod 341 = 56 \neq n-1$

$a^{2^1 d} \bmod n = 38^{170} \bmod 341 = 67 \neq n-1$.

Since none of the congruences hold, we know 341 is composite. In fact, $341 = 11 \cdot 31$. However, consider $n = 703$ and $a = 3$. $703 - 1 = 702 = 2^1 \cdot 351$. So $s = 1$ and $d = 351$. Continuing:

$a^d \bmod n = 3^{351} \bmod 703 = 702 \neq 1$

$a^{2^0 d} \bmod n = 3^{351} \bmod 703 = 702 = n-1$

By the second congruence, 703 is a strong pseudoprime base 3. If we then try $a = 5$, we get:

$a^d \bmod n = 5^{351} \bmod 703 = 438 \neq 1$

$a^{2^0 d} \bmod n = 5^{351} \bmod 703 = 438 \neq n-1$.

This time neither congruence holds, and we know 703 is a composite number. In fact, $703 = 19 \cdot 37$.

The Miller-Rabin test is very fast and has a complexity of $O\left((\log n)^3\right)$. Of course, because it is probabilistic, there is a chance of the test returning a number as prime when it is in fact composite. However, as will be demonstrated later, we are very concerned with the speed of the primality test and no deterministic test will run fast enough for our purpose. The Miller-Rabin test offers us both speed, as compared to other primality tests, and the ability to control the probability of error and will be our tool of choice.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DIFFIE-HELLMAN AND THE DISCRETE LOGARITHM

A. THE DIFFIE-HELLMAN PROTOCOL

"We stand today on the brink of a revolution in cryptography." This was the first sentence in a breakthrough paper published in 1977 by Whitfield Diffie and Martin E. Hellman. In the paper, titled *New Directions in Cryptography* [10], the authors introduced the idea of public key cryptography and a key exchange protocol that was named in their honor. The Diffie-Hellman protocol provided the first practical solution to the key distribution problem, allowing two parties, never having met in advance or shared keying material, to establish a shared secret by exchanging messages over an open channel. The key can then be used to encrypt subsequent communications using a symmetric key cipher. The security rests on the intractability of the Diffie-Hellman problem and the related problem of computing discrete logarithms [1]. We will call the two parties conducting the key exchange "Alice" and "Bob."

Protocol steps:

1. A prime number p and generator α of $\mathbb{Z}_p^*(2 \le \alpha \le p-2)$ are selected and published.
2. Alice chooses a random secret $x, 1 \le x \le p-2,$ and sends Bob $\alpha^x \bmod p$

   $A \to B : \alpha^x \bmod p$
3. Bob chooses a random secret $y, 1 \le y \le p-2,$ and sends Alice $\alpha^y \bmod p$

   $B \to A : \alpha^y \bmod p$
4. Bob receives α^x and computes the shared key as $K = (\alpha^x)^y \bmod p$
5. Alice receives α^y and computes the shared key as $K = (\alpha^y)^x \bmod p$

Because $(\alpha^y)^x = (\alpha^x)^y$, Alice and Bob have arrived at the same secret key. Only x, y, and α^{xy} are kept secret. All other values are sent in the clear. The example below illustrates the procedure.

1. Alice and Bob agree on $p = 37$ and $\alpha = 2$.
2. Alice chooses $x = 14$ and sends Bob $30 (\equiv 2^{14} \bmod 37)$.

$$A \rightarrow B : 30$$

3. Bob chooses $y = 23$ and sends Alice $5 (\equiv 2^{23} \bmod 37)$.

$$B \rightarrow A : 5$$

4. Bob receives 30 and computes $30^{23} \bmod 37 = 28$
5. Alice receives 5 and computes $5^{14} \bmod 37 = 28$

Alice and Bob have agreed upon 28 as their secret key.

Figure 2 demonstrates which parties know what information. The man-in-the-middle will be called Eve from here on out.

Alice	
Knows	*Does not Know*
p=37	y=23
α=2	
x=14	
α^x=30	
α^y=5	
$(\alpha^y)^x$=K=28	

Bob	
Knows	*Does not Know*
p=37	x=14
α=2	
y=23	
α^x=30	
α^y=5	
$(\alpha^x)^y$=K=28	

Eve	
Knows	*Does not Know*
p=37	x=14
α=2	y=23
α^x=30	K=28
α^y=5	

Figure 2. Diffie-Hellman Example

Obviously, a much larger value of p is required than used in the example to make the key agreement potentially secure. If the prime number 37 was used, Eve could simply try all possible values of $2^{xy} \bmod 37$. Because 2 is a primitive root modulo 37, this can take 36 values. A key space with only 36 possibilities can be exhausted with ease. However, if the prime number used is large enough, no computing power available today can exhaust the key space. For instance, most applications recommend 1024-bit primes [2]. This correlates to a number of about 300 digits and makes searching the key space one by one infeasible. Table 4 demonstrates how long it would take a modern personal computer (PC) and a super-computer (SC) to exhaust various sizes of key spaces. We assume a PC can search approximately one million (10^6) keys per second, while a super-computer can search approximately one trillion (10^{12}) keys per second.

For instance, if a prime of 64 bits was used, it would correlate to a base-ten number of approximately 19 digits. The key space would be all the numbers $1, 2, ..., p-1$, which would be on the order of 10^{19} numbers. Therefore, a PC would take $\frac{10^{19}}{10^6} = 10^{13}$ seconds to completely search the entire key space.

Bits	Digits (approximate)	PC time (approximate)	SC time (approximate)
64	19	317,098 years	115 days
128	39	3 x 10^(25) years	3 x 10^(19) years
256	77	3 x 10^(63) years	3 x 10^(57) years
512	154	3 x 10^(140) years	3 x 10^(134) years
1024	308	3 x 10^(294) years	3 x 10^(288) years
2048	616	3 x 10^(602) years	3 x 10^(596) years

Table 4. Times to Exhaust a Key Space

Considering most applications use prime of 1024 bits or greater, it is obviously infeasible to conduct a random search of an entire key space. Of course, one could get lucky and the key could be one of the first numbers searched by the computer. However, as indicated by the enormous times listed in the table, it is more likely a random key search would take longer than most scientists believe the universe has existed.

B. THE DISCRETE LOGARITHM

Eve has more information than just the fact that the key resides in the interval $(1, p-1)$. Because the exchange occurs over an open channel, Eve knows α^x and α^y as well. If $\beta \equiv \alpha^x \pmod{p}$ and $\gamma \equiv \alpha^y \pmod{p}$, then p, α, β, and γ are known. All Eve has to do is solve $\alpha^x \equiv \beta \pmod{p}$ for x or $\alpha^y \equiv \gamma \pmod{p}$ for y. Once x or y are known, Eve simply raises α^x to y or α^y to x and arrives at the secret key K. However, if p is large, solving $\alpha^x \equiv \beta \pmod{p}$ for x in general is considered difficult. The problem of finding x in this case is known as the ***discrete logarithm problem*** (DLP), often abbreviated $x = L_\alpha(\beta)$.

The difficulty of solving the DLP yields useful cryptosystems. Diffie-Hellman key exchange protocol, El Gamal encryption system, and the Digital Signature Algorithm all rely on the difficulty of solving the DLP. However, not all public-key crypto systems rely on the difficulty of the DLP. Another number theory problem that yields cryptosystems is the problem of factoring large integers. RSA, considered by many to be the most popular public-key cryptography algorithm, relies on the difficulty of factorization for its security. The size of the largest primes for which discrete logs can be computed has usually been approximately the same size as the size of largest integers that could be factored [11]. In 2005, a 168 digit prime (556 bits) discrete logarithm was computed, setting a record at that time. The record factorization up to then was 200 digits (663 bits).

As discussed above, if p is small, it is easy to compute discrete logs by exhaustive search. However, when is p large, this is not feasible. We will now discuss several methods of attacking the DLP.

1. The Pohlig-Hellman Algorithm

Pohlig and Hellman introduced the following algorithm in 1978 to solve discrete logs when $p-1$ has only small prime factors [11], [12].

Suppose

$$p-1=\prod_i q_i^{r_i}$$

is the factorization of $p-1$ into primes. Let q^r be one of the factors. The idea is to compute $x \pmod{q^r}$ for each $q_i^{r_i}$ and combine them using the Chinese Remainder Theorem to find the discrete logarithm.

Thus, $x \pmod{q^r}$ is found by writing $x = x_0 + x_1 q + x_2 q^2 + \ldots$ with $0 \le x_i \le q-1$ and determining the coefficients $x_0, x_1, \ldots, x_{r-1}$.

General idea: Starting with $\beta \equiv \alpha^x$, raise both sides to the $\frac{p-1}{q}$ to obtain

$$\beta^{(p-1)/q} \equiv \alpha^{x(p-1)/q} \equiv \alpha^{x_0(p-1)/q}(\alpha^{p-1})^{x_1 q + x_2 q^2 + \ldots} \equiv \alpha^{x_0(p-1)/q} \pmod{p}$$

To find x_0, simply look at the powers

$$\alpha^{k(p-1)/q} \pmod{p}, \quad k = 0, 1, 2, \ldots q-1,$$

until one of them yields $\beta^{(p-1)/q}$. Then $x_0 = k$.

An extension of this idea yields the remaining coefficients. Assume $q^2 \mid p-1$. Let

$$\beta_1 \equiv \beta\alpha^{-x_0} \equiv \alpha^{q(x_1 + x_2 q + \ldots)} \pmod{p}$$

Raise both sides to the $\frac{p-1}{q^2}$ power to obtain

$$\beta_1^{(p-1)/q^2} \equiv \alpha^{(p-1)(x_1 + x_2 q + \ldots)/q} \equiv \alpha^{x_1(p-1)/q}(\alpha^{p-1})^{x_2 + x_3 q + \ldots} \equiv \alpha^{x_1(p-1)/q} \pmod{p}.$$

To find x_1, simply look at the powers

$$\alpha^{k(p-1)/q} \pmod{p},\ k = 0,1,2,...q-1,$$

Until one of them yields $\beta_1^{(p-1)/q^2}$. Then $x_1 = k$.

If $q^3 \mid p-1$, let $\beta_2 \equiv \beta_1 \alpha^{-x_1 q}$, and raise both sides to the $\frac{p-1}{q^3}$ power and find x_2.

We can continue this process until we find that q^{r+1} does not divide $p-1$. We have then determined $x_0, x_1 ... x_{r-1}$, so we know $x \pmod{q^r}$.

Repeat this procedure for all prime factors of $p-1$. This yields $x \pmod{q^r}$ for each $q_i^{r_i}$ and we combine these using the Chinese Remainder Theorem to find $x \pmod{p-1}$. Since $0 \le x < p-1$, this determines x.

As an example, let us solve $2^x \equiv 3 \pmod{101}$ for x.

$$p-1 = 100 = 2^2 \cdot 5^2 \text{ so } q = 2,5$$

First, we solve $2^x \equiv 3 \pmod{2^2}$. Let $x = x_0 + 2x_1 \pmod{2^2}$. Then

$$\beta^{(p-1)/2} \equiv 3^{50} \equiv -1 \pmod{101} \text{ and } \alpha^{(p-1)/2} \equiv 2^{50} \equiv -1 \pmod{101}$$

So $-1 \equiv (-1)^{x_0}$ and $x_0 = 1$.

Continuing, $\beta_1 \equiv \beta\alpha^{-x_0} \equiv 3 \cdot 2^{-1} \equiv 3 \cdot 51 \equiv 52 \pmod{101}$. So $\beta_1^{(p-1)/2^2} \equiv 52^{25} \equiv 1 \pmod{101}$ and $1 \equiv (-1)^{x_1}$. So $x_1 = 0$ and $x \equiv 1 + 2 \cdot 0 \equiv 1 \pmod{2^2}$.

Next, we solve $2^x \equiv 3 \pmod{5^2}$. Let $x = x_0 + 5x_1 \pmod{5^2}$. Then

$$\beta^{(p-1)/5} \equiv 3^{20} \equiv 84 \pmod{101} \text{ and } \alpha^{(p-1)/5} \equiv 2^{20} \equiv 95 \pmod{101}$$

We make a list,

$$95^0 \equiv 1; 95^1 \equiv 95; 95^2 \equiv 36; 95^3 \equiv 87; 95^4 \equiv 84 \pmod{101}.$$

Matching with the list, we see that $x_0 = 4$.

Continuing, we get $\beta_1 \equiv \beta\alpha^{-x_0} \equiv 3 \cdot 2^{-4} \equiv 3 \cdot 19 \equiv 57$. So $\beta_1^{(p-1)/5^2} \equiv 57^4 \equiv 87 \pmod{101}$. We again compare with the above list and see that $95^3 \equiv 3$ and $x_1 = 3$. This leads to $x \equiv 4 + 5 \cdot 3 \equiv 19 \pmod{5^2}$.

Now, we combine $x \equiv 1 \pmod{2^2}$ and $x \equiv 19 \pmod{5^2}$ using the Chinese Remainder Theorem to find $x = 69$. So $2^{69} \equiv 3 \pmod{101}$.

It is well known that the time complexity of the Pohlig-Hellman algorithm is $O(\sqrt{p})$ [11].

2. Baby Step, Giant Step

Eve is trying to solve $\alpha^x \equiv \beta \pmod{p}$ for *x*. The following algorithm was developed by Daniel Shanks [11].

First, choose an integer N with $N^2 \geq p-1$. Next, make two lists:

1. $\alpha^j \bmod p$ for $0 \leq j < N$

2. $\beta\alpha^{-Nk} \bmod p$ for $0 \leq k < N$

Look for a match between the two lists. If one is found, then $\alpha^j \equiv \beta\alpha^{-Nk}$, so $\alpha^{j+Nk} \equiv \beta$. Therefore, $x = j + Nk$ and the discrete logarithm is solved.

The complexity of the baby step, giant step algorithm is also $O(\sqrt{p})$, but it requires storing approximately $\sqrt{p}$ numbers in memory and is therefore, impractical for very large primes, such as 10^{20} or larger [11].

3. The Index Calculus

Again, Eve is trying to solve $\alpha^x \equiv \beta \pmod{p}$ for x. The idea in the index calculus method is similar to the quadratic sieve method of factoring [11].

The first step is a precomputation step and involves picking a factor base and searching for a set of r linearly independent relations between the factor base and the powers of α. Let B be a bound and let $p_1, p_2, ..., p_m$ be the primes less than B. This is our factor base. We then compute $\alpha^k \pmod{p}$ for r values

of k. For each number, try to write it as a product of the factor base. If this is not the case, discard α^k. However, if $\alpha^k = \prod p_i^{a_i} \pmod p$, then

$$k \equiv \sum a_i L_\alpha(p_i) \pmod{p-1}.$$

When we obtain enough relations, we can solve for $L_\alpha(p_i)$ for each i.

Next, for random integers s, compute $\beta\alpha^s \pmod p$. For each such number, try to write it as a product of primes less than B. If we succeed, we have $\beta\alpha^s \equiv \prod p_i^{b_i} \pmod p$, which means

$$L_\alpha(\beta) \equiv -s + \sum b_i L_\alpha(p_i) \pmod{p-1}.$$

Using this algorithm, any p over 200 digits will be difficult to solve, which makes the Index Calculus good only for moderate-sized primes [11]. One can show that the time complexity of the Index Calculus is $O(e^{c(\ln n)^{1/3}(C\ln\ln n)^{2/3}})$ for some $c > 0$, if implemented by the Number Field Sieve.

C. THE DIFFIE-HELLMAN PROBLEM

We described how solving the discrete logarithm easily would allow Eve to arrive at the secret key. There is another problem Eve can solve to arrive at the secret key—namely, the Diffie-Hellman Problem. The Diffie-Hellman Problem comes in two flavors, the computational and the decisional. The ***Computational Diffie-Hellman Problem*** is defined as follows: Let p be a prime and let α be a primitive root $\bmod p$. Given $\alpha^x \pmod p$ and $\alpha^y \pmod p$, find $\alpha^{xy} \pmod p$. Recall that Eve has access to both α^x and α^y as they are both made public during the exchange. It is not currently known whether or not this problem is easier than computing discrete logs [11]. A related problem, known as the ***Decisional Diffie-Hellman Problem***, is defined as follows: Let p be a prime and let α be a primitive root $\bmod p$. Given $\alpha^x \pmod p$ and $\alpha^y \pmod p$, and $\beta \neq 0 \pmod p$, decide whether or not $K \equiv \alpha^{xy} \pmod p$ [11]. In other words, if someone offers a number to Eve and claims it is K, can Eve decide whether or not that person is

telling the truth with the information captured in the open channel? Like the computational Diffie-Hellman problem, the decisional Diffie-Hellman problem has yet to be solved. It is unknown whether a method for solving the decisional problem will lead to a solution for the computational problem.

The methods described for solving discrete logarithms above force applications that rely on the difficultly of solving discrete logs to stay away from certain primes. Obviously, the larger the prime used, the better. Baby-step Giant-step and the Index Calculus become infeasible to use when primes are larger than 200 digits. The Pohlig-Hellman algorithm relies on the factorization of $p-1$ to consist of only small primes. If p does not contain only small primes, the algorithm becomes inefficient. Therefore, the primes chosen when using the Diffie-Hellman protocol should contain at least one large prime in the factorization of $p-1$. This situation gives rise to the attack we will focus on. If $p-1$ contains a very large prime, such that $p-1=Rq$ with q prime and R a small integer, an unauthenticated exchange becomes vulnerable to an active man-in-the-middle attack that we will discuss next.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MAN-IN-THE-MIDDLE ATTACK

A. THEORY BEHIND THE ATTACK

Wiener and van Oorschot [2] noted that, if certain primes are used, a potentially fatal protocol attack on the Diffie-Hellman key exchange protocol becomes possible. The idea is based on forcing the parties to agree on a shared key that resides in a subgroup of the cyclic group $\mathbb{Z}_p^*$. If the order of the subgroup is small enough, an adversary can exhaustively search the subgroup, retrieve the secret key, and eavesdrop on the communication of Alice and Bob.

For instance, consider the case when the prime used for the key exchange is of the form $p = 2q + 1$, where q is a prime. Then, $\alpha^q = \alpha^{(p-1)/2}$.

Claim: $\alpha^{(p-1)/2}$ is an element of order two.

Proof: By Fermat's little theorem, $\alpha^{p-1} = 1 \mod p$. So $\alpha^{(p-1)/2}$ must be +1 or -1. But if $\alpha^{(p-1)/2} = 1$ then α must have order $(p-1)/2$. This is a contradiction, because α is a primitive root of $\mathbb{Z}_p^*$ and must be of order $p-1$. So $\alpha^{(p-1)/2} = -1$ and is an element of order two. □

If Alice and Bob respectively send each other unauthenticated messages α^x and α^y, an active intruder may substitute $(\alpha^x)^q$ for the first, and $(\alpha^y)^q$ for the second. When Alice receives $(\alpha^y)^q$ and computes $(\alpha^{qy})^x$ and when Bob receives $(\alpha^x)^q$ and computes $(\alpha^{qx})^y$, they will arrive at only one of two possible values, +1 and -1. The intruder can then try both possible keys and gain access to Alice and Bob's secret communications. Obviously, if Alice and Bob demonstrate vigilance, they will agree in advance to suspect any key agreement that arrives at +1 or -1.

We can generalize the situation if Alice and Bob use a prime number of the form $p = Rq + 1$, where R is a small integer and q is again a large prime.

Claim: $\alpha^{(p-1)/R}$ is an element of order R.

Proof: Raising $\alpha^{(p-1)/R}$ to consecutive powers, starting with 0, we get:

$$(\alpha^{(p-1)/R})^0 = 1,\ (\alpha^{(p-1)/R})^2,\ (\alpha^{(p-1)/R})^3, \ldots, (\alpha^{(p-1)/R})^R = \alpha^{(p-1)} = 1.$$

This produces a list of R different values. Continuing after R,

$$(\alpha^{(p-1)/R})^{(R+1)} = (\alpha^{(p-1)/R})^R \cdot (\alpha^{(p-1)/R})^1 = 1 \cdot (\alpha^{(p-1)/R}),$$

$$(\alpha^{(p-1)/R})^{(R+2)} = (\alpha^{(p-1)/R})^R \cdot (\alpha^{(p-1)/R})^2 = 1 \cdot (\alpha^{(p-1)/R})^2, \ldots\ldots,$$

$$(\alpha^{(p-1)/R})^{(R+n)} = (\alpha^{(p-1)/R})^R \cdot (\alpha^{(p-1)/R})^n = 1 \cdot (\alpha^{(p-1)/R})^n$$

For $n < R$, the results are in the original list.

For $n \geq R$, we can write $R + n = R + kR + m$ with $0 \leq m \leq R-1$ and $m, k \in \mathbb{Z}$.

$$(\alpha^{(p-1)/R})^{(R+n)} = (\alpha^{(p-1)/R})^{(R+kR+m)} = (\alpha^{(p-1)/R})^R \cdot (\alpha^{(p-1)/R})^{kR} \cdot (\alpha^{(p-1)/R})^m =$$

$$= 1 \cdot 1^k \cdot (\alpha^{(p-1)/R})^m = (\alpha^{(p-1)/R})^m$$

Because $0 \leq m \leq R-1$, this is in our original list and $\alpha^{(p-1)/R}$ is of order R. □

So, if the prime Alice and Bob agree to use is of the form $p = Rq + 1$, Eve can force them to agree on a key in a subgroup of $\mathbb{Z}_p^*$ of order R by replacing α^x and α^y with $(\alpha^x)^q$ and $(\alpha^y)^q$. Even if Alice and Bob are vigilant, the key can take any of R values and the generalized attack poses a significant threat to an unauthenticated key exchange using the Diffie-Hellman protocol.

B. CREATING THE ENVIRONMENT

Eve must force Alice and Bob into a subgroup of small order to conduct this attack. Figure 3 represents a possible algorithm Eve could follow.

NOTE: Eve only needs to consider cases when R is even, because if R is odd, $\frac{p-1}{R}$, must be even and cannot be prime. Also, if Eve calculates $\frac{p-1}{m}, m \in \mathbb{Z}$ as a non-integer, she can obviously ignore trying any number of the form $\frac{p-1}{km}, k \in \mathbb{Z}$ because it will also not be an integer.

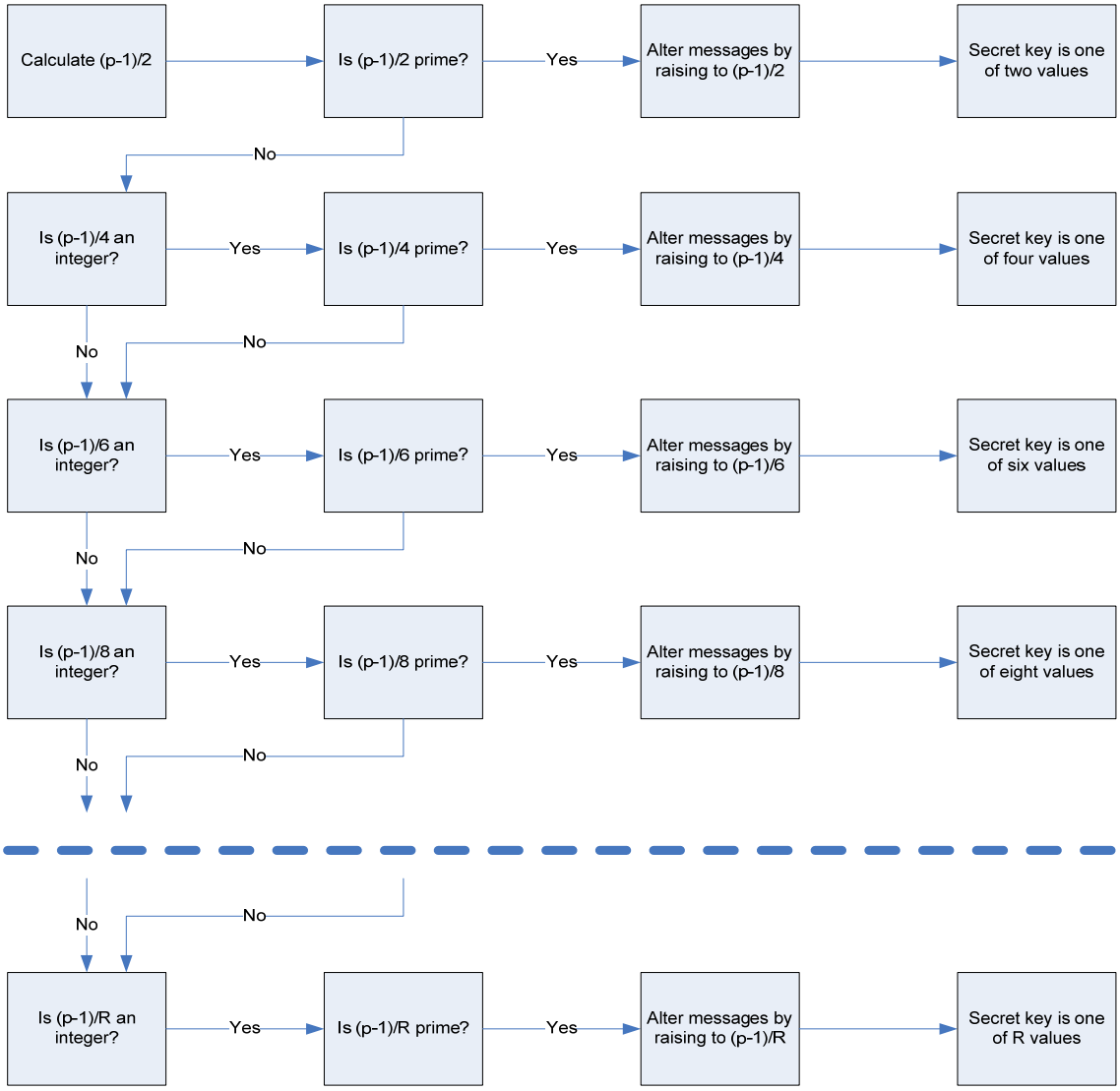

Figure 3. Attack Algorithm

The most important step in creating the environment to conduct the attack is searching $\frac{p-1}{k}$, $k = 2, 4, ..., R$, until we find a prime. We cannot continue the attack until we find such a prime. Obviously, the longer Alice and Bob are kept waiting for return correspondence, the more suspicious they will become of possible compromise of their communication. Therefore, we need the fastest

possible method to detect primality. From our discussion in Chapter II, we know a probabilistic primality test suits us best. Specifically, we could use the Miller-Rabin primality test with complexity $O\left((\log n)^3\right)$.

If we are forced to search the entire index k from $2 \le k \le R$, how long might this take us? Recall that we only need try even values of k, and in the worst case, we may be forced to try all $R/2$ even numbers. Therefore, the worst case scenario in searching for a prime would take

$$\frac{\sum_{k=2}^{R} O\left((\log N)^3\right)}{2} = O\left((R/2)\cdot(\log N)^3\right) = O\left((\log N)^3\right)$$

steps, with N being the input number into the Miller-Rabin primality test. Thus the constant value in the Big-O estimate changes, but the algorithm remains bounded by the time it takes to conduct the primality tests.

As an example, suppose Eve was listening to Alice and Bob agree upon the prime number to use for their key exchange to take place in the near future. The prime number they choose is $p = 10007$ with a primitive root of $\alpha = 3$. Eve uses the attack algorithm in Figure 4 to attempt to force Alice and Bob to agree to a key in a subgroup of $\mathbb{Z}_{10007}^{*}$.

First, $\frac{p-1}{2} = \frac{10006}{2} = 5003$

Next, Eve runs 5003 through the Miller-Rabin primality test and the result is prime.

This situation represents the initial case described above with the prime number being of the form $p = 2q+1$. Specifically, $10007 = 2\cdot 5003 + 1$. Next, Eve must intercept the number Alice attempts to send to Bob. Suppose Alice chooses $x = 758$ and attempts to send $\alpha^x (3^{758} \bmod 10007 \equiv 4865)$ to Bob.

$$A \to E : 4865$$

Eve intercepts the communication, then takes $\alpha^x(4865)$ and raises it to the q power.

$$(\alpha^x)^q \equiv (3^{758})^{5003} \bmod 10007$$

Meanwhile, Eve must also intercept the number Bob is attempting to send to Alice. Suppose Bob chooses $y = 555$ and attempts to send $\alpha^y(3^{555} \bmod 10007 \equiv 1771)$ to Alice.

$$B \to E : 1771$$

Eve again intercepts the communication, and takes $\alpha^y(1771)$ and raises it to the q power.

$$(\alpha^y)^q \equiv (3^{555})^{5003} \bmod 10007$$

Eve then sends the results to the intended recipients.

$$E \to B : 4865^q \bmod 10007$$

$$E \to A : 1771^q \bmod 10007$$

Alice and Bob then both finish the key agreement by raising the received number to their private keys, x and y respectively, and arrive at the same number, the “secret” key.

$$(\alpha^{yq})^x \equiv (\alpha^{xq})^y$$

As a result of the theory discussed above, without any knowledge of x or y, Eve knows the only possible keys are 1 and 10006. Eve must wait for a message to be sent between Alice and Bob, try both keys, and figure out which one is being used. She can then eavesdrop, and Alice and Bob’s secret communication has been compromised.

However, as mentioned before, any vigilance on the part of Alice or Bob would cause suspicion if the key agreed upon were of the form +1 or -1.

Now, suppose the prime number Alice and Bob agreed upon was $p = 19991$ and $a = 3$. Eve must again search for a large prime factor of $p-1$.

First, $\frac{p-1}{2} = \frac{19990}{2} = 9995$

Next, Eve would run 9995 through the Miller-Rabin primality test. However, because it ends with a five, five must be a factor and it cannot be a prime number.

Continuing, $\frac{p-1}{4} = \frac{19990}{4} = 4997.5$ is not an integer.

$\frac{p-1}{6} = \frac{19990}{6} = 3333.\overline{66}$ is not an integer.

Because $\frac{p-1}{4}$ was not an integer, we skip $\frac{p-1}{8}$.

$\frac{p-1}{10} = \frac{19990}{10} = 1999$

Next, Eve runs 1999 through the Miller-Rabin primality test and the result is prime.

Eve has found a large prime factor of $p-1$. This situation resembles the generalized attack with a prime of the form $p = Rq + 1$; in this case $19991 = 10 \cdot 1999 + 1$. Intercepting, altering, and retransmitting the messages as she did above, Eve again forces Alice and Bob into a subgroup of the original cyclic group. This time, however, there are ten possibilities for the "secret" key.

$$\left(\alpha^{\frac{p-1}{R}}\right)^1, \left(\alpha^{\frac{p-1}{R}}\right)^2, ..., \left(\alpha^{\frac{p-1}{R}}\right)^R =$$

$$\left(3^{1999}\right)^1, \left(3^{1999}\right)^2, ..., \left(3^{1999}\right)^{10}$$

The cyclic subgroup of $\mathbb{Z}^*_{19991}$ generated by 3^{1999} is of order ten and Alice and Bob can only arrive at ten values for their key. Eve must wait for Alice and Bob to communicate with their new key and see which of the ten values Alice

and Bob agreed on. Once a message is intercepted, Eve can pull it offline, attempt each possible key, determine the key they agreed upon, and listen in on Alice and Bob's communication.

C. PRIMES OF THE FORM $Rq+1$

For this man-in-the-middle attack to be possible, Alice and Bob must agree to choose a prime of the form $Rq+1$. How likely is it, assuming Alice and Bob are using random large primes, that the prime they choose will be of the correct form? To answer this question, we must first count the number of primes p, such that $p=Rq+1$. We can begin with the case where $R=2$. This represents the original case in the man-in-the-middle attack, where $p=2q+1$. These particular prime numbers have their own name. A prime p is a so-called ***Sophie Germaine*** (SG) prime if $2p+1$ is also prime. If we let $\pi_{SG}(t)$ be the number of SG primes not exceeding t, it can be demonstrated that

$$\pi_{SG}(t)=O\left(\frac{t}{(\log t)^2}\right) \quad [13]$$

Now, considering the general case, if we fix R, then the number of primes $p\le t$ of the form $p=Rq+1$ is

$$=O\left(\frac{t}{\phi(R)(\log(t/R))^2}\right)$$

where $\phi(t)$ is Euler's Phi function [14]. However, in the attack R can range from 2 to some bound, say B. Therefore, we must sum the cases from $R=2$ to $R=B$. The number of primes p such that $p=Rq+1$ with q prime, ranging from $2\le R\le B$ with $B\le t^{1/2}$ is

$$=O\left(\frac{t}{(\log t)^2}\right)\sum_{R\le B}\frac{1}{\phi(R)}$$

$$=O\left(\frac{t\log B}{(\log t)^2}\right) \quad [14]$$

The ***prime number theorem*** states that, if $\pi(x)$ is the prime counting function, then $\lim_{x\to\infty}\frac{\pi(x)}{x/\ln(x)}=1$. Roughly speaking, this tells us that if you randomly select a number close to a large number N, the odds of it being prime are about $1/\ln(N)$. By the prime number theorem, it follows that $\lim_{x\to\infty}\frac{\pi_{SG}(x)}{\pi(x)}=0$. If we let $\pi_{Rq+1}(t)$ count the number of primes of the form $p=Rq+1$ not exceeding t, it follows that $\lim_{x\to\infty}\frac{\pi_{Rq+1}(x)}{\pi(x)}=0$ as well. This tells us that, as x gets very large, the likelihood that a random prime number is a Sophie Germaine Prime or any prime of the form $Rq+1$ is increasingly unlikely.

Using the prime number theorem and Big-O estimates above with a constant value of one, we can approximate the numbers of primes of different forms. Table 5 lists these approximations using scientific notation. The R value corresponds to different values for primes of the form $p=Rq+1$. The ratios listed are: (primes of the given form) / (total primes).

	0-64 bits	64-128 bits	128-256 bits
Total Primes	4.1583e17	3.8353e36	6.5255e74
R=2 (S.G)	9.3737e15 *ratio: .0225*	4.3228e34 *ratio: .0113*	3.6775e72 *ratio: .0056*
R=100	4.316e16 *ratio: .1038*	1.9907e35 *ratio: .0519*	1.6935e73 *ratio: .0260*
R=10^4	8.6335e16 *ratio: .2076*	3.9815e35 *ratio: .1038*	3.3871e73 *ratio: .0519*
R=10^6	1.295e17 *ratio: .3114*	5.9722e35 *ratio: .1557*	5.0806e73 *ratio: .0779*
	256-512 bits	**512-1024 bits**	**1024-2048 bits**
Total Primes	3.778e151	2.5327e305	2.2765e613
R=2 (S.G.)	1.0646e149 *ratio: .0028*	3.5683e302 *ratio: .0014*	1.6037e610 *ratio: .0007*
R=100	4.9024e149 *ratio: .0130*	1.6433e303 *ratio: .0065*	7.3853e610 *ratio: .0032*
R=10^4	9.8049e149 *ratio: .0260*	3.2865e303 *ratio: .0130*	1.477e611 *ratio: .0065*
R=10^6	1.4707e150 *ratio: .0389*	4.9298e303 *ratio: .0195*	2.2156e611 *ratio: .0097*

Table 5. Prime Number Approximations

The approximations demonstrate the increasing unlikelihood of a random prime being of the form $p = Rq + 1$. Using our approximations, around 64 bits over 30% of all primes match the form with a bound of 10^6. However, when we consider primes around 2048 bits, the percentage drops below one. If we increase the bound we can increase the likelihood, but increasing the bound forces the attacker to search through more keys to find the correct one.

D. COUNTERMEASURES AGAINST THE ATTACK

To prevent this potentially fatal protocol attack, Alice and Bob have several options. The easiest method is to force authentication prior to the key exchange. Another method that prevents the attack is based on creating a prime order subgroup before the key exchange takes place.

1. Authentication

The attack we have discussed is not the only man-in-the-middle attack Diffie-Hellman is vulnerable to. The Appendix details another attack, if no authentication occurs prior to the key exchange. To combat these attacks, a variation of Diffie-Hellman that ensures authentication can be used. An example of such a variation is the Station-to-Station protocol (STS). STS is a three-pass variation of the basic Diffie-Hellman protocol that allows the establishment of a shared secret key between two parties with mutual entity authentication and mutual explicit key authentication [1]. The STS employs digital signatures. A digital signature of a message is a number dependent on some secret known only to the signer; and, additionally, on the content of the message being signed [1]. The STS protocol is frequently employed with the RSA signature scheme.

To employ an RSA signature scheme, public and private key pairs must first be generated.

RSA signature scheme key generation steps [1]:

1. Generate two large distinct random primes p and q, each roughly the same size
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$
3. Select a random integer $e, 1 < e < \phi$, such that $\gcd(e, \phi) = 1$
4. Use the extended Euclidean algorithm to compute the unique integer $d, 1 < d < \phi$ such that $ed \equiv 1 \pmod{\phi}$
5. The user's public key is (n, e) and the user's private key is d

NOTE: Each user should generate a public and private key

Now, if a user Alice wants to sign a message m, and a user Bob wants to verify the message signature, the remaining steps of the protocol must be completed.

RSA signature scheme protocol steps [1]

1. Signature generation
    a. Compute $\tilde{m} = R(m)$, an integer in the range $[0, n-1]$
    b. Compute $s = \tilde{m}^d \bmod n$
    c. Alice's signature for m is s.
2. Signature verification
    a. Obtain Alice's authentic public key (n, e)
    b. Compute $\tilde{m} = s^e \bmod n$
    c. Recover $m = R^{-1}(\tilde{m})$

With the knowledge of a digital signature scheme, in particular RSA, we can move onto the STS protocol. If we let E denote a symmetric encryption algorithm, and $S_A(m)$ denote Alice's signature on m, the protocol is as follows [1]:

1. Set up
    a. A prime number p and generator α of $\mathbb{Z}_p^*(2 \le \alpha \le p-2)$ are selected and published
    b. Alice selects RSA public and private signature keys (n_A, e_A), and d_A (Bob selects analogous keys). Assume each party has access to authentic copies of the other's public key.
2. Actions
    a. Alice generates a secret random $x, 1 \le x \le p-2$ and sends to Bob $\alpha^x \bmod p$.

$$A \to B : \alpha^x \bmod p \text{ (message 1)}$$

b. Bob generates a secret random $y, 1 \le y \le p-2$, and computes the shared key $k = (\alpha^x)^y \bmod p$. Bob signs the concatenation of both exponentials, encrypts this using the computed key, and sends to Alice.

$$B \to A : a^y \bmod p, E_k(S_B(\alpha^y, \alpha^x)) \text{ (message 2)}$$

c. Alice computes the shared key $k = (\alpha^y)^x \bmod p$, decrypts the encrypted data, and uses Bob's public key to verify the received value as the signature on the hash of the cleartext exponential received and the exponential sent in message 1. Upon successful verification, Alice accepts that k is actually shared with Bob, and sends Bob an analogous message.

$$A \to B : E_k(S_A(\alpha^x, \alpha^y)) \text{ (message 3)}$$

d. Bob similarly decrypts the received message and verifies Alice's signature therein. If successful, Bob accepts that k is actually shared with Alice.

The exchanged exponentials are digitally signed and retransmitted during the STS protocol. Therefore, Eve cannot alter the original exponentials without triggering a failure during Alice and Bob's key agreement. This precludes the man-in-the-middle attack we have focused on and defends Alice and Bob's key exchange against several other possible active man-in-the-middle attacks.

2. Prime Order Subgroups

Van Oorschot and Wiener [2] noticed the potentially fatal man-in-the-middle attack and reasoned that restricting computations to prime-order subgroups would prevent the attack. In this case, we will force the prime number p that defines the environment to be of the form $p = Rq + 1$, where R is a small

integer and q is a large prime. Now, instead of using a generator α of $\mathbb{Z}_p^*$ as our base for exponentiation, we compute $g = \alpha^{(p-1)/q}$ and let g be our new base.

Claim: The element g generates a subgroup of order q.

Proof: Suppose g is of order $k < q$ and so $g^k \equiv 1$. Then $\alpha^{(p-1)\cdot k/q} \equiv 1$. But $k/q < 1$ and so $(p-1)\cdot k/q < (p-1)$. This means α is of order $< (p-1)$, a contradiction because α is a generator of $\mathbb{Z}_p^*$. Therefore, g must be of order $\geq q$. But $g^q = \alpha^{(p-1)\cdot q/q} = \alpha^{(p-1)} \equiv 1$, so g is of order q and $\langle g \rangle$ is an subgroup of order q. □

By using g instead of α to conduct the key exchange, Alice and Bob are working in a prime order subgroup instead of a group of order $p-1$. The man-in-the-middle attack we have discussed is based on forcing the parties into a subgroup of small order and exhaustively searching the smaller key space. However, by Lagrange's theorem, the order of any subgroup must divide the order of the group. The order of the group generated by g is q. Therefore, any subgroup must be of order q or 1, because those are the only divisors of q. Thus, the prime order subgroup cannot be divided any further and this man-in-the-middle attack becomes infeasible.

The Internet Engineering Task Force (IETF) has adopted the prime order subgroup tactic to prevent the type of attack we have focused on. In particular, Request for Comment (RFC) 2631 standardizes the technique for a particular Diffie-Hellman variant, based on the American National Standards Institute x9.42 draft [15].

E. EXTENDING THE ATTACK TO THE N-PARTY SETTING

The Diffie-Hellman protocol we have discussed so far has been limited to two parties. However, protocols have been created that extend the key agreement to group communications. Steiner, Tsudik and Wainer [16] defined a

class of natural extensions of Diffie-Hellman to the n-party setting. These protocols, without the countermeasures discussed above, are vulnerable to the man-in-the-middle attack we have focused on. We now move to demonstrate the attack on two of the protocols the authors describe. First, we consider the protocol the authors name Group Diffie-Hellman version 1 (GDH.1). In this section, to keep with the original notation of [16], we use set notation to mean an ordered tuple.

We call the participants of the n-party key exchange $\{M_1, M_2, ..., M_n\}$. As in the two-party case, a prime number p and a generator α of the group Z_p^* are selected and published. Each member M_i chooses a random secret number $s_i, 0 < s_i \leq p-2$. The protocol consists of two stages; upflow and downflow.

In the upflow stage, each member makes their contribution to the shared key. A member M_i receives a collection of intermediate values, and has the task of raising the last in the list of incoming intermediate values to the power of s_i. Then M_i appends the result to the incoming set of values and forwards all to M_{i+1}. As an example, M_3 would receive $\{\alpha^{s_1}, \alpha^{s_1 s_2}\}$ from M_2. M_3 would then compute $\alpha^{s_1 s_2 s_3}$, append the result to the incoming message to create $\{\alpha^{s_1}, \alpha^{s_1 s_2}, \alpha^{s_1 s_2 s_3}\}$ and forward to M_4.

The upflow stage is completed when M_n calculates $\alpha^{s_1 s_2 \cdots s_n}$, which is the intended group key, K_n. Once M_n has obtained K_n, the downflow stage is initiated. Each member M_i receives i messages, one to compute K_n and $i-1$ to send to M_{i-1}. For example, if $n=4$, M_3 would receive $\{\alpha^{s_4}, \alpha^{s_1 s_4}, \alpha^{s_1 s_2 s_4}\}$ from M_4. First, M_3 would use the last value to compute $K_n = \alpha^{s_1 s_2 s_3 s_4}$. Then, the remaining values would be raised to s_3 and $\{\alpha^{s_4 s_3}, \alpha^{s_1 s_4 s_3}\}$ would be sent to M_2.

M_2 would repeat the procedure, and would send $\left\{\alpha^{s_4 s_3 s_2}\right\}$ to M_1. The downflow stage is then completed when M_1 computes $K_n = \alpha^{s_1 s_2 s_3 s_4}$. GDH.1 is depicted in Figure 4.

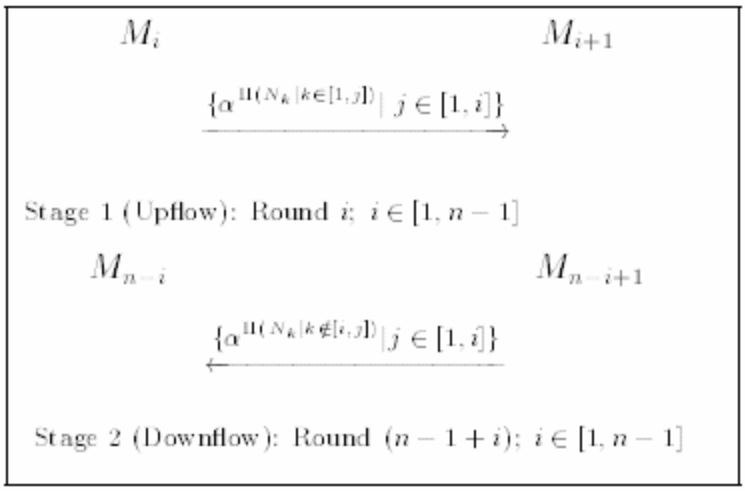

Figure 4. GDH.1 [From 16]

The active adversary, Eve, wishes to attack the key agreement forcing the n-party to agree on a key in a small subgroup of Z_p^*. Like in the two-party case, if possible Eve must first break the prime number p down into the form $p = Rq + 1$ with q a large prime and R a small integer. Once completed, Eve must then intercept and alter two messages to complete the attack. The first message she must intercept is the first message sent, that is,

$$M_1 \rightarrow M_2 : \alpha^{s_1}.$$

With α^{s_1} captured, Eve computes $(\alpha^{s_1})^q = \alpha^{qs_1}$ and proceeds to send the computed number as the message onto M_2. M_2 computes $\alpha^{qs_1 s_2}$ and sends $\left\{\alpha^{qs_1}, \alpha^{qs_1 s_2}\right\}$ to M_3. This continues until the end of the upflow stage, when M_n computes $\alpha^{qs_1 \ldots s_n} = K_n$. Eve has forced K_n to be one of R values, based on the theory of the attack described earlier in the chapter.

Next, Eve must intercept the first message sent during the downflow stage. If $n=4$, then

$$M_4 \rightarrow M_3 : \left\{ \alpha^{s_4}, \alpha^{qs_1s_4}, \alpha^{qs_1s_2s_4} \right\}.$$

NOTE: Because of the alteration Eve completed in the upflow stage, only the first part of the message must be altered.

Eve simply computes α^{qs_4}, replaces the first number with the computation, and forwards the message to M_3. The participants all arrive at $K_n = \alpha^{qs_1s_2s_3s_4}$, and the key exchange has been successfully attacked. However, in this case Eve had to capture and alter two very specific messages for the attack to be successful. In the next protocol, Eve has more flexibility.

Next, we turn our attention to Group Diffie Hellman version 3 (GDH.3). GDH.3 reduces the amount of computation each party (except for M_n) must complete, which may be very beneficial if the group size is large. The protocol consists of four stages. The first stage is similar to the upflow stage of GDH.1 in which every member contributes to the key. However, after processing the upflow message, M_{n-1} broadcasts $\alpha^{s_1s_2\ldots s_{n-1}}$ to the entire group as the second stage of the process. In stage three, each M_i, except M_n, factors out their contribution (α^{s_i}) from the broadcasted value and forwards the result to M_n. After M_n collects all the values from the group, in the last stage M_n raises each value to s_n and returns the values to the group. Now each M_i has $\alpha^{\prod s_k | k\in[1,n], k\neq i}$ and simple raises this value to s_i to compute K_n.

For example, if $n=5$, the upflow stage completes when M_4 computes $\alpha^{s_1s_2s_3s_4}$. Then, in stage 2, this value is broadcasted to the entire group. In stage 3, each member other than M_5 factors out their contribution and forwards the result to M_5 (i.e. M_2 would send $\alpha^{s_1s_3s_4}$). In stage 4, M_5 raises each received

value to s_5 and returns the value to the sender (i.e. M_2 would receive $\alpha^{s_1 s_3 s_4 s_5}$. Lastly, each member raises the received value to their secret number and arrives at K_n. Figure 5 depicts GDH.3.

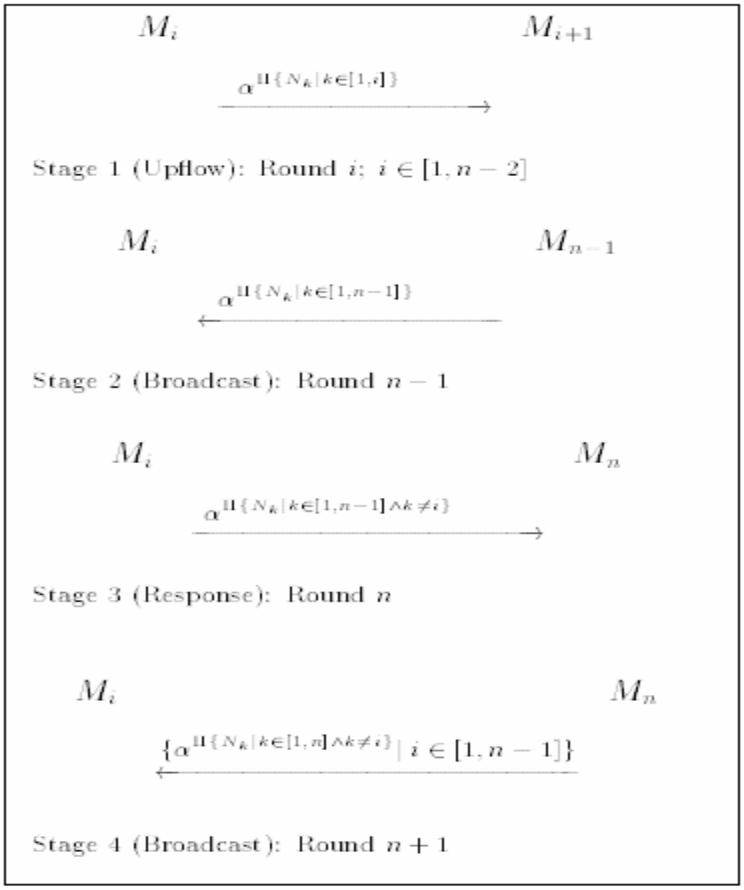

Figure 5. GDH.3 [From 16]

It is much easier for Eve to attack GDH.3 than GDH.1. She needs only to intercept and alter one message, and she can choose any of the first $i-2$ messages sent in the group. By raising any one of these messages to q, M_{n-1} will inevitably broadcast $\alpha^{qs_1 s_2 \ldots s_{n-1}}$ to the group. At this point, each member factors out their contribution, and forwards the result to M_n leaving q in the exponent of each message sent. M_n simply raises each message to s_n and returns each message. Therefore, q is undisturbed, each member arrives at the same key $K_n = \alpha^{qs_1 s_2 \ldots s_n}$, and Eve has successfully forced the group into a small number of

possible values for the key. However, as mentioned above, if the parties agree to use either authentication or prime order subgroups during the key exchange, attacks of this sort are prevented.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS AND FUTURE WORK

This thesis investigated and analyzed a particular man-in-the-middle attack on the Diffie-Hellman key exchange protocol. We created an algorithm to carry out the attack and demonstrated how it is constrained by the primality test used by the attacker. In particular, if the Miller-Rabin primalty test is used, the algorithm's complexity is $O\left((\log N)^3\right)$ with N being the input prime number. We showed that prime numbers of the form $p = Rq + 1$ with R bounded are common with small primes but become increasingly rare as larger numbers are considered. In fact, with low bit primes such as 128 bits, a reasonably-sized R will give an attacker a good chance of the prime being of the desired form. However, when large primes such as 1024 and 2048 bits are considered, a very large value of R is required to give an attacker a reasonable chance of conducting the attack. We demonstrated how two techniques, authentication and prime order subgroups, can prevent the attack. In fact, it appears industry has begun to adopt the prime order subgroup technique to defend against the attack. Finally, we demonstrated how the attack can be expanded to include a class of multi-party Diffie-Hellman variants.

Possible future efforts include coding and implementing the man-in-the-middle attack on active communications to test the theory laid out in this thesis. It is possible that analyzing the given prime number, capturing the required messages, altering those messages, and forwarding the messages to the intended recipients will be too time-consuming. This would obviously alert the parties of possible compromise. In addition, it may be possible to alter the attack to compromise communications that are authenticated and render several Diffie-Hellman variants such as the STS protocol vulnerable. Other future work may include an attempt to defeat the prime order subgroup technique.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: ANOTHER MAN-IN-THE-MIDDLE ATTACK

This appendix details a possible man-in-the-middle attack on the Diffie-Hellman key exchange protocol, if no prior authentication occurs [17].

1) Alice sends her public key to Bob, but Eve intercepts it, and Bob never receives the key.

2) Eve spoofs Alice's identity and sends over her public key to Bob. Bob now thinks that he has Alice's public key.

3) Bob sends his public key to Alice, but Eve intercepts it, and Alice never receives the key.

4) Eve spoofs Bob's identity and sends over her public key to Alice. Alice now thinks that she has Bob's public key.

5) Alice combines her private key and Eve's public key and creates symmetric key S1.

6) Eve combines her private key and Alice's public key and creates symmetric key S1.

7) Bob combines his private key and Eve's public key and creates symmetric key S2.

8) Eve combines her private key and Bob's public key and creates symmetric key S2.

9) At this point, Alice and Eve share a symmetric key (S1) and Bob and Eve share a different symmetric key (S2). Alice and Bob think they are sharing a key between themselves and do not realize that Eve is involved.

10) Alice writes a message to Bob, uses her symmetric key (S1) to encrypt the message, and sends it.

11) Eve intercepts the message and decrypts it with the symmetric key S1, reads or modifies the message and re-encrypts it with symmetric key S2, and sends it to Bob.

12) Bob takes symmetric key S2 and uses it to decrypt and read the message.

Figure 6 illustrates the attack [17].

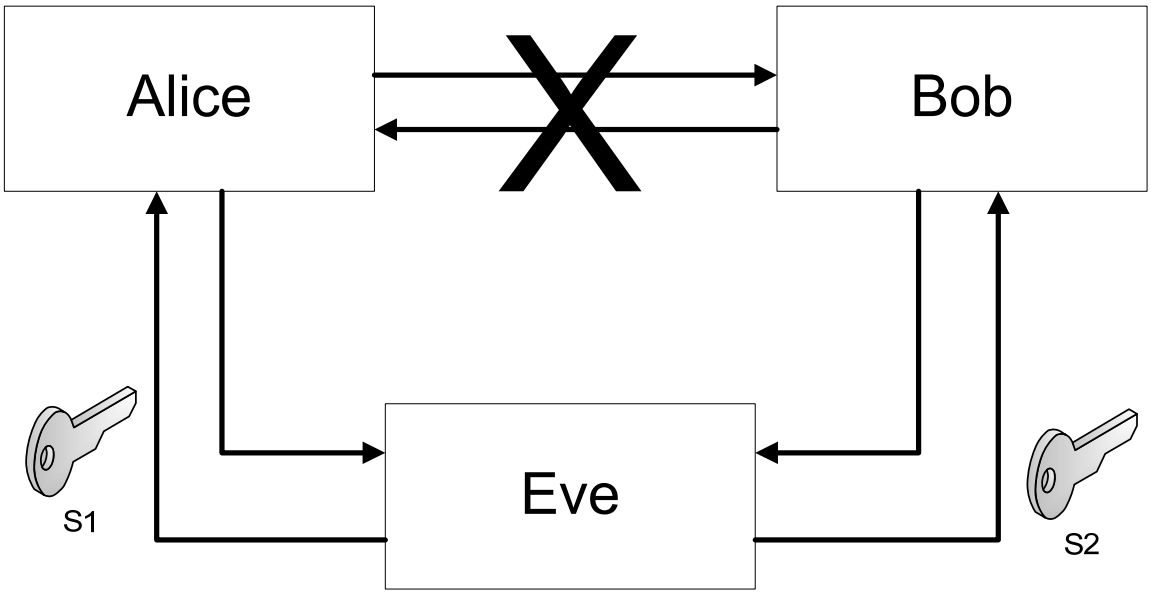

Figure 6. Another Man-in-the-Middle Attack

LIST OF REFERENCES

[1] A. J. Menezes, P. C. van Oorshot, and S. A Vanstone, *Handbook of Applied Cryptography*. CRC Press, New York, New York, 1997.

[2] P. C. van Oorschot and M. J. Wiener, On Diffie-Hellman Key Agreement with Short Exponents. *EUROCRYPT'96*, LNCS 1070, Springer-Verlag, 1996, pp. 332–343.

[3] S. Singh, *The Code Book*. Doubleday, 1999.

[4] J. B. Fraleigh, *A First Course in Abstract Algebra.* Addison-Wesley, San Francisco, CA, 7th Edition, 2002.

[5] K. H. Rosen, *Discrete Mathematics and Its Applications*. McGraw Hill, San Francisco, CA, 6th Edition, 2007.

[6] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*. Springer, New York, NY, 2001.

[7] A. L. Atkin and F. Morain, Elliptic Curves and Primality Proving. *Res. Rep.* 1256, INRIA, June 1990.

[8] M. Agrawal, N. Kayal, N. Saxena, PRIMES is in P. *Annals of Mathematics 160.* 2004.

[9] C. Pomerance and H.W. Lenstra, Primality testing with Gaussian periods, preprint.

[10] W. Diffie and M. E. Hellman, *New Directions in Cryptography.* IEEE IT-22, 1976, pp. 644–654.

[11] W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory.* Pearson, Upper Saddle River, NJ, 2nd Edition, 2006.

[12] S. Pohlig and M. Hellman, An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. *IEEE Transactions on Information Theory,* 24, 1978, pp. 106–110.

[13] T. Agoh, On Sophie Germain Primes. *Tatra Mt. Math.* Publ. 20, 2000.

[14] P. Stanica, private communication, 2009.

[15] Internet Engineering Task Force (IETF) Request for Comment (RFC) 2631, June 1999.

[16] M. Steiner, G. Tsudik, and Mi. Waidner, Diffie-Hellman Key Distribution Extended to Group Communication. *3rd ACM Conference on Computer and Communications Security*. March 14–16 1996, New Delhi, India.

[17] S. Harris, *All In One CISSP.* McGraw-Hill, San Francisco, CA, 2005.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California