Int. J. Math. And Appl., 8(2)(2020), 121–131
ISSN: 2347-1557
Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

# Applications of Number Theory to Cryptography: A Look Into the Diffie-Hellman Key Exchange and the RSA Cipher

**Suraj Oruganti**[1,*]

1 West-Windsor Plainsboro High School South, 346 Clarksville Rd, Princeton Junction, New Jersey, United States.

**Abstract:** Number theory is the study of integer values, and as Carl Friedrich Gauss once coined it, "the queen of mathematics." More specifically, a field of study in number theory is the behavior of prime numbers. Cryptography is the development of algorithms in order to transmit information securely. Number Theory is intertwined with, and is an integral part of cryptography. Therefore, in this paper we will be discussing the applications of number theory to cryptography.

**Keywords:** Cryptography, Diffie-Hellman Key Exchange, RSA Cipher



## 1. Introduction

Number theory has many practical applications in the real world. For example, a simple situation: Alpha and Beta are brothers and they love to play the game Detective. Alpha wants to send Beta a very important message that only Beta is allowed to read. However, their evil sister named Gamma sits right between them, so they will have to pass the message along through her. Since Alpha only wants Beta to decipher the message, Alpha tells Beta during lunch: *I have a clever way to outsmart Gamma. I am going to change every letter of my message with the previous letter in the alphabet, and I will change the letter 'a' to 'z'. To be able to read the information I send you, change each letter with the next letter in the alphabet and replace any letter 'z' with the letter 'a'.*

During lunch, Alpha writes the following message: *Fzllz rjhoodc rbgnnk!*. He gives the sheet to Gamma, and tells him that it is for Beta and to pass it. Gamma agrees, however, she takes a quick look at the sheet, but doesn't understand the gibberish written on it. Beta receives the information and using the rule Alpha told him on how to change the letters, the message reads *Gamma skipped school!*.

The encryption process involves the changing of the original message(also known as the plaintext) *Gamma skipped school!* to the encrypted message(also known as the ciphertext) *Fzllz rjhoodc rbgnnk!*. The decryption process involves the changing of the encrypted message *Fzllz rjhoodc rbgnnk!* to the original message *Gamma skipped school!*

Cryptography is the scientific method of protecting information and communication through the use of algorithmic-based codes. We use cryptography in order to make sure information is transmitted securely. The above example was a very simple display of how basic cryptography works. A brief overview of the subject can be found here: [5]. Applications of

* *E-mail: surajoruganti@gmail.com*

cryptography include privacy of data, web browsing and secure communications(which include financial transactions and email). Without cryptography, we wouldn't have much security over our private information!

This paper is concerned with the uses of modern cryptographic algorithms, namely the Rivest-Shamir-Adleman algorithm(RSA) and the Diffie-Hellman Key Exchange. The RSA is the most widely used algorithm of the 20th century, but coupling the Diffie-Hellman Key Exchange with other crytographic ciphers, such as the Discrete Logarithm Problem, has promising uses in our expanding technological world. In order to understand these complex algorithms, a strong number theory background is necessary. These number theory concepts are outlined in Part 1 of this paper. Part 2 will highlight a discussion about the uses and the strengths of the before mentioned ciphers.

An additional note: The familiar computer language Java is used throughout the text to come up with computer programs that perform certain algorithms. We include our Java codes for all our algorithms in this article in boldface. Note that we only use Java functions or methods that are in the Java database, or ones that were user created.

## 2. GCD: The Rules of Divisibility

**Definition 2.1.** *Let $m$ and $n$ be two integers. The statements $m$ divides $n$, and $m$ is a divisor of $n$, are equal if there exists an integer $x$ such that $n = mx$.*

For example, 5 divides 10 because $10 = 5(2)$.

**Proposition 2.2.** *Let $m$ and $n$ be two integers. There exists a unique integer $q$ and a unique non-negative integer $r$ such that $m = qn + r$ and $r < |n|$. $q$ is the **quotient** and $r$ is the **remainder** when $n$ is divided by $m$.*

For instance, if we divide 47 by 13, $47 = 3(13) + 8$ and $0 \leq 8 < 13$. This means our quotient $q = 3$, and our remainder $r = 8$. It can be noted that $m$ divides $n$ is equivalent to the remainder $r$ of $n$ divided by $m$ being equal to 0. This is because we get the form $m = qn$ when $r$ is 0, which is equivalent to our definition of divisibility.

There are Java commands for computing the quotient and remainder after the division of numbers $m$ and $n$. This is executed with the syntax $m//n$ and $m\%n$ accordingly. As an illustration, $10//3 = 3$ and $10\%3 = 1$. Feel free to try some of these commands out on a Python compiler.

**Notation 2.3.** *Let $m$ and $n$ be integers. The quotient and the remainder of $m$ divided by $n$ are denoted in this article by $m//n$ and $m\%n$ respectively.*

**Definition 2.4.** *Given that $m_1$, $m_2$, .... $m_n$ is a set of integers, the greatest common divisor of the set is denoted by $\gcd(m_1, m_2, ....m_n)$. This notation shows the largest positive number that divides all of these integers.*

It follows that the greatest common divisor of 49 and 14 is 7. According to our notation, $\gcd(49, 14) = 7$. Note that two numbers $m$ and $n$ are **relatively prime** if $\gcd(m, n) = 1$. We can now say this:

**Proposition 2.5.** *For all integers $m$ and $n$:*

$$\gcd(m, n) = \gcd(n, m\%n). \tag{1}$$

Here is the proof of the above definition:

Let $\gcd(m, n) = g$. Note that by Fact 1, we have that $m = qn + r$. Additionally, $m\%n = r$. Now, we know that $g$ divides both $m$ and $n$. Therefore, we know that $g$ divides $m - qn$, meaning that $g$ divides $r$. $g$ must be the largest possible common factor of $r$ and $n$, because any other common factor would also have to divide $m$. However, $g$ is the **greatest** common

factor. Therefore, we know that $g = \gcd(n, r) = \gcd(n, m\%n)$, as desired. For instance, if $m = 40$ and $n = 15$, it follows that $m\%n = 40\%15 = 10$. Equation 1 now holds, because $\gcd(40, 15) = \gcd(15, 10) = 5$.

**Proposition 2.6.** *If there exists two integers $m$ and $n$, then there must also exist integers $a$ and $b$ such that*

$$\gcd(m, n) = am + bn. \tag{2}$$

The numbers $a$ and $b$ are not pairwise unique. There indeed exist infinitely many pairs of numbers that we could pick for $a$ and $b$. One example of such a pair is $\gcd(40, 15) = 5$ and $5 = (2)40 + (-5)15$.

# 3. The Euclidean Algorithm

**Definition 3.1.** *Let $m$ and $n$ be two positive integers. We have that:*

*(1). Let $r_0 = m$.*

*(2). Let $r_1 = n$.*

*(3). Let the index $i = 1$.*

*(4). Let $r_{i-1} = q_i r_i + r_{i+1}$, where $q_i$ is a random integer, such that $0 \leq r_{i+1} < r_i$*

*(5). If $r_{i+1} = 0$, then $\gcd(m, n) = r_i$.*

*(6). If $r_{i+1} > 0$, then replace $i$ with $i + 1$ and repeat the sequence from Step IV.*

For example, if $m = 40$ and $n = 15$, then $r_0 = 40$, $r_1 = 15$, $r_2 = 40\%15 = 10$, $r_3 = 15\%10 = 5$, and $r_4 = 10\%5 = 0$. We can now generalize this fact:

**Proposition 3.2.** *Let $r_0, r_1, \ldots, r_c$ be the sequence earlier referenced, where $c$ is the positive integer such that $r_{c-1} \neq 0$ and $r_c = 0$. Then $\gcd(m, n) = r_{c-1}$.*

The defined Java function headed as **gcd**, takes two integers $m$ and $n$ and returns the greatest common divisor.

```
public static int gcd(int m, int n)

    int rp = m;

    int rc = n;

    while(rc != 0)

        int i = rp;

        rp = rc;

        rc = rc % i;
```

```
return rp;
```

Now, let $m$ and $n$ be two positive integers. Let us reference our sequence $r_0, r_1, \ldots, r_c$. We define sequences $a_0, a_1, \ldots, a_c$ and $b_0, b_1, \ldots, b_c$ as follows:

$$a_0 = 1, a_1 = 0, \qquad \text{for } 1 \leq i < c, \qquad a_{i+1} = a_{i-1} - (r_{i-1}//r_i)a_i \tag{3}$$

$$b_0 = 0, b_1 = 1, \qquad \text{for } 1 \leq i < c, \qquad b_{i+1} = b_{i-1} - (r_{i-1}//r_i)b_i. \tag{4}$$

We have that $a_0m + b_0n = r_0$ because $a_0m + b_0n = 1r_0 + 0r_1 = r_0$. Note that $a_1m + b_1n = r_1$ because $a_1m + b_1n = 0r_0 + 1r_1 = r_1$. In fact, we can make the following conjecture:

**Proposition 3.3.** *We have $a_im + b_in = r_i$, and $0 \leq i < n$. Additionally, since $r_{n-1} = \gcd(m, n)$, we have*

$$a_{n-1}m + b_{n-1}n = \gcd(m, n). \tag{5}$$

We can now define another function **gcdr** that two integers as an input, $m$ and $n$ and returns three integers $g, a, b$, such that $g = \gcd(m, n)$ and $am + bn = g$. Additionally, note that // is not the function to find a divisor in Java, we simply use / as a form of concatenation.

```
public static int[] gcdr(int m, int n)

    int rp = m;

    int rc = n;

    int g = gcd(m,n);

    int[] array = new int[3];

        int i = g - m*a;

        if(i%n == 0)

            int b = i / n;

            array[0] = a;

            array[1] = b;

            array[2] = g;

            break;
```

**return array;**

Using our earlier, **gcdr**$(40, 15)$ returns the three numbers $2, -5, and 5$.

# 4. Prime Factorization

**Definition 4.1.** *A positive integer $p \geq 2$ is said to be prime if and only if the positive integers that divide $p$ are 1 and $p$ itself.*

For example, 11 is prime, since only 1 and 11 itself divide 11. However, 15 is not prime, since it is also divisible by 3 and 5, neither of which are equal to 1 or 15.

**Definition 4.2.** *A positive integer $p \geq 2$ is said to be composite if it is not prime.*

Since 15 is not a prime number, we call 15 a **composite** number.

**Proposition 4.3.** *The prime factorization of an integer $n$ is said to be a unique finite sequence of primes $p_1 < p_2 < \ldots < p_k$ and a unique finite sequence of positive $n_1, n_2, \ldots, n_k$ such that $n = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$.*

For example, $24 = 2^3 * 3$, $27 = 3^3$, and $3 = 3$ are a few examples of how the prime factorization of specific numbers is written. Additionally, it is noteworthy that the prime factorization of a prime number is simply itself. We can also define a function known as Euler's totient function that is the following:

**Definition 4.4.** *Let $n$ have a prime factorization of $n = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$. Then, we can denote $\phi(n)$ as:*
$\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \ldots (1 - \frac{1}{p_k})$

This function actually counts the number of integers less than or equal to $n$ that are coprime to $n$.

# 5. Modular Congruences

**Definition 5.1.** *For integers $m$, $n$, and $a$, we can denote $m \equiv n \pmod{a}$ if $a$ divides $m - n$. We say that $m$ is congruent $n$ modulus $a$. If $0 \leq n < a$, then $n$ is the remainder when $m$ is divided by $a$.*

For instance, 19 is congruent 4 modulus 5, i.e. $19 \equiv 4 \pmod{5}$ because 5 divides $19 - 4 = 15$. Additionally, since $0 \leq 4 < 5$, we know that 4 is the remainder when 19 is divided by 5. Indeed, it is, because $19 = 5(3) + 4$. This is further generalized below, in a slightly different form:

**Proposition 5.2.** *For any two integers, $m$ and $n$, we have:*

$$m \equiv m\%n \pmod{n}. \tag{6}$$

For example, $20 \equiv 20\%6 \equiv 2 \pmod{6}$.

**Proposition 5.3.** *Let $a$, $b$, $c$, $d$ and $m$ be an integers such that $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$. Then,*

$$a + c \equiv b + d \pmod{m} \quad and \quad ac \equiv bd \pmod{m}. \tag{7}$$

For example, $17 \equiv 2 \pmod{5}$, $12 \equiv 2 \pmod{5}$ and $29 = 17 + 12 \equiv 4 = 2 + 2 \pmod{5}$ and $204 = 17(12) \equiv 4 = 2(2) \pmod{5}$.

## 6. Solution to $an \equiv b \pmod{m}$

We start with integers $a$, $b$ and $m$ and $m > 0$. We want to prove the following:

**Proposition 6.1.** *We have an integer $y = \gcd(a, m)$, and it divides $b$ evenly. $i$ and $j$ are integers existing such that $ia + jm = y$. If $n = (ib/y)\%m$, then $an \equiv b \pmod{m}$ and $1 \leq n < m$.*

The proof is shown below:

$$an = a((ib/y)\%m) \equiv (aib/y + jmb/y) \equiv (b/y)(ai + jm) \equiv b \pmod{m}.$$

Now, we accordingly define **axcm** that inputs $a$, $c$ and $m$. Note that the starting condition is $\gcd(a, m)$ divides $c$, and returns $x$ the only solution of $ax \equiv c \pmod{m}$ with $1 \leq x < m$.

```
public static int axcm(int a, int c, int m)

    int[] array = gcdr(a,m);

    int d = array[0];

    int u = array[1];

    int v = array[2];

    if(d != 0)

        return (u*(c/d))%m;

    else

        return 0;
```

For example, **axcm**$(5, 4, 17)$ returns the the integer 11, and this is true because $5x \equiv 4 \pmod{17}$, $5(11) - 4 = 51 = 17(3)$ and thus, 17 divides $5(11) - 4$.

## 7. Euler's Theorem

**Proposition 7.1.** *Consider $p$ and $q$ to be two prime integers and $m = pq$. Then, there exists an $a$, such that $a^{(p-1)(q-1)} \equiv 1 \pmod{m}$ and where $\gcd(a, m) = 1$.*

Euler's theorem is stated by the following:

**Proposition 7.2.** *Consider $n$ to be a positive integer and $a$ be a positive integer relatively prime to it. We must have the following:*

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Note that $\phi(pq) = pq(1-\frac{1}{p})(1-\frac{1}{q}) = (p-1)(q-1)$, and Fact 10 follows accordingly. To demonstrate Euler's Theorem, we use $p = 3$ and $q = 7$. On Table 1, we show all the powers of $a^k$ modulus 21 for $1 \leq k \leq 12$, and for all $a$ such that $\gcd(a, 21) = 1$ and $1 \leq a < 21$. Notice that $a^{(p-1)(q-1)} = a^1 2 \equiv 1 \pmod{21}$ for all such numbers $a$.

| $a$ | $a^2$ | $a^3$ | $a^4$ | $a^5$ | $a^6$ | $a^7$ | $a^8$ | $a^9$ | $a^{10}$ | $a^{11}$ | $a^{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 8 | 16 | 11 | 1 | 2 | 4 | 8 | 16 | 11 | 1 |
| 4 | 16 | 1 | 4 | 16 | 1 | 4 | 16 | 1 | 4 | 16 | 1 |
| 5 | 4 | 20 | 16 | 17 | 1 | 5 | 4 | 20 | 16 | 17 | 1 |
| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 |
| 10 | 16 | 13 | 4 | 19 | 1 | 10 | 16 | 13 | 4 | 19 | 1 |
| 11 | 16 | 8 | 4 | 2 | 1 | 11 | 16 | 8 | 4 | 2 | 1 |
| 13 | 1 | 13 | 1 | 13 | 1 | 13 | 1 | 13 | 1 | 13 | 1 |
| 16 | 4 | 1 | 16 | 4 | 1 | 16 | 4 | 1 | 16 | 4 | 1 |
| 17 | 16 | 20 | 4 | 5 | 1 | 17 | 16 | 20 | 4 | 5 | 1 |
| 19 | 4 | 13 | 16 | 10 | 1 | 19 | 4 | 13 | 16 | 10 | 1 |
| 20 | 1 | 20 | 1 | 20 | 1 | 20 | 1 | 20 | 1 | 20 | 1 |

**Table 1.** **Table of powers of $a$ modulus 21 for all $a < 21$ such that gcd$(a, 15) = 1$.**

# 8. Solution to $a^k \equiv b \pmod{m}$

**Proposition 8.1.** *Consider $k$, $b$ and $m$ be three integers that satisfy the following:*

1. *$b$ and $m$ are relatively prime.*
2. *There exist $p$ and $q$ that are prime numbers such that $m = pq$*
3. *$k$ and $(p-1)(q-1)$ are relatively prime.*

*There exist integers $i$ and $j$ such that $ik + j(p-1)(q-1) = 1$. Then, the solution that satisfies $1 \leq a < m$ is*

$$x = (b^i)\%m. \tag{8}$$

It is noteworthy that $a^k = ((b^i)\%m)^k \equiv ((b^i))^k \pmod{m} \equiv (b^{ik}) \pmod{m}$:

$$i^k \equiv b^{ik} \pmod{m}. \tag{9}$$

From Proposition 10, we have that $b^{(p-1)(q-1)} \equiv 1 \pmod{m}$, therefore $1 \equiv 1^j \equiv (b^{(p-1)(q-1)})^j \equiv b^{j(p-1)(q-1)} \pmod{m}$:

$$1 \equiv b^{j(p-1)(q-1)} \pmod{m}. \tag{10}$$

By multiplying the two equations above, it follows that $a^k \equiv b^{ik} b^{j(p-1)(q-1)} \equiv b^{ik+j(p-1)(q-1)} \equiv b^1 \equiv b \pmod{m}$. The function **xkbm** takes $k, b, p, q, and m$ as an input, and returns as output the integer $x$ that satisfies $x^k \equiv b \pmod{m}$ and $1 \leq x < m$.

```
public static int xkbm(int b, int k, int p, int q, int m)

    int array[] = gcdr(k, (p-1)*(q-1));
```

```
int d = array[0];

int u = array[1];

int v = array[2];

return (Math.pow(b,u))%m;
```

We have now learned all of the necessary number theory tools to understand how our crytpographic ciphers will function. We will now examine the RSA cipher, the Discrete Logarithm Problem and the Diffie-Hellman Key Exchange.

# 9. RSA Cipher

Now, let us consider a scenario with our old friends, Alpha and Beta. Alpha wants to send a more complicated encoded message to Beta, because Gamma figured out their original pattern relatively easily. Beta will decrypt it once he receives the message. The following algorithm is used:

(1). Right before transmitting the message, they select two large prime numbers $p$ and $q$. The numbers $p$ and $q$ are considered a part of the **private domain**, meaning that only Alpha and Beta know the identity of $p$ and $q$.

(2). Alpha chooses a $k$ accordingly to satisfy $\gcd(k,(p-1)(q-1))=1$, and a $m$, such that $m=pq$. The numbers $k$ and $m$ are considered a part of the **public domain**, meaning that everyone knows the identity of $k$ and $m$.

(3). Alpha wants to transmit the message $r<m$ to Beta, which is a string of numbers. However, he instead computes $f=(r^k)\%m$ and send $f$ to Beta.

(4). Beta knows the identity of $p$, $q$ and $k$, and since $\gcd(k,(p-1)(q-1))=1$, he can solve for $n$ the the equation $n^k \equiv f \pmod{m}$ with $1 \leq n < m$. But $r^k \equiv f \pmod{m}$ and $1 \leq r < m$. There is only one possible value for $n$, so we have that $n=r$, and Beta can successfully decode the message.

The values of $p$ and $q$ are required to decipher the equation $n^k \equiv f \pmod{m}$ with $1 \leq x < m$. Beta is the only one(other than Alpha) that has access to private domain, therefore Beta is the only one that can fully perform the above algorithm. These set of steps that Alpha and Beta have used is known as the Rivest-Shamir-Adleman cipher, or more commonly known as the RSA cipher. It has many applications in the real world, including in email, virtual private network(VPN), and even chat apps!

This method **enc** encrypts the message:

```
public static int enc(s,k,m)
    return (Math.pow(s,k)%m);
```

This method **dec** that decrypts the message:

```
public static int dec(k,e,p,q)
    return xkbm(k,e,p,q);
```

# 10. The Discrete Logarithm Problem

In this article we describe the theory behind the Discrete Logarithm Problem(DLP). We will be using the number theory concepts developed in the previous chapters to discuss DLP. Before we jump right into our discussion of DLP, there is a bit of background behind the conditions. For instance, the idea of groups, and more specifically, cyclic groups. Consider the following definition:

**Definition 10.1.** *A group is a set $G$, such that there is an operation $*$, such that for any two elements $a$ and $b$ in our set $G$, we perform $a * b$ and add it to $G$. There are 3 conditions which much be satisfied for our operation $*$ :*

*(1). There exists an identity element $c$, such that $c * a = a * c = a$.*

*(2). The operation is associative.*

*(3). For every element, there exists a respective inverse in the set, such that the operation$(*)$ between this inverse and the original element is the identity element.*

Knowing what a group is, we can also define a cyclic group:

**Definition 10.2.** *A cyclic group $G$ is a group in which there exists an element of the group $a$, such that for every element $b$ of $G$, we have that $b = a^n$, for some integer $n$.*

Finally, let us define what the generator of a certain group is:

**Definition 10.3.** *A generator of a certain set $G$ is an element $g$, such that any element of the group can be written of the form $ng$, where $n$ is any integer.*

The generator is a primitive root. We are now ready to declare our DLP:

**Definition 10.4.** *We are given some cyclic group $G$. Let the generator of group $G$ be denoted as $g$. Let $a$ be any random element of $G$. We need to find a number $n$, such that $a \equiv g^n \pmod{p}$.*

Note that $n$ is referred to as the discrete logarithm of $a$ with respect to the base $g$. Now, how do we solve the DLP? There is a straightforward way of solving it:
If we are able to find the $g$ that is our generator, and the number $ng$, then we can obviously find $n$, since $n = (ng) * g^{-1}$. $g^{-1}$ is simply the inverse of $g$ in the respective multiplicative group. However, finding the certain $g$ and $ng$ that work in a modulus gets significantly harder when dealing with larger primes. That is why we need to couple DLP with the Diffie-Hellman Key Exchange, which we will introduce in the next chapter.

# 11. The Diffie-Hellman Key Exchange

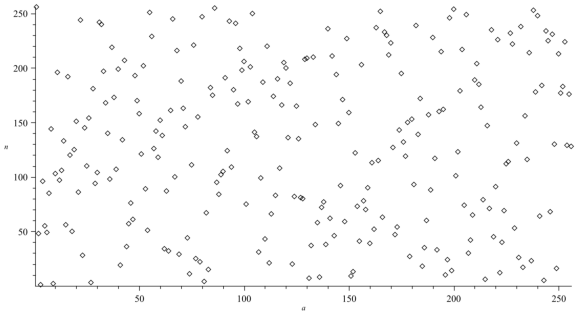
The Diffie-Hellman Key Exchange works as follows:

**Definition 11.1.** *A prime number $p$ is decided among the two parties, as is a primitive root of $p$, namely $g$. Note that anybody can access $p$ and $g$. Next, they each pick two random numbers $a$ and $b$, which is private information. One of the parties computes $g^a \pmod{p}$, and the other one computes $g^b \pmod{p}$. They send it to each other. Then, they each raise it to the $b$th and $a$th powers respectively, and obtain $g^{ab} \pmod{p}$, and this number is the key.*

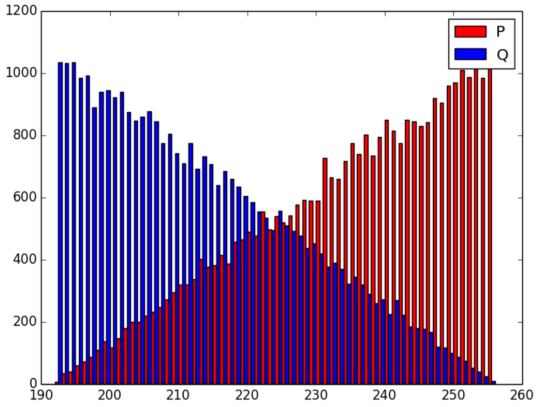
For instance, let us assume Alpha and Beta agree on $p = 293$ and $g = 34$. Alpha will choose the secret key of $a = 128$ and Beta will choose the secret key of $b = 245$. Alpha computes $34^{128} \equiv 137 (\mathrm{mod}\, 293)$. Beta computes $34^{245} \equiv 130 (\mathrm{mod}\, 293)$. They send each other these newly computed numbers, and then they both end up computing:
$130^{128} \equiv 137^{245} \equiv 191 (\mathrm{mod}\, 293)$. 191 is the key.
Now obviously, this is very easy to compute with a computer, which is why the difficulties of cracking the Diffie-Hellman key exchange start when the prime number modulus is very large, around 100 bits!

## 12. Discussion: Putting it All Together

When looking at the Diffie-Hellman Key Exchange and the RSA cipher, there are clear similarities. For instance, both are the most widely-used forms of encryption and security today. As far as the performance of each set of ciphers goes, both a standard 1024-bit Diffie-Hellman key and a 1024-bit RSA key would have an equal efficiency. The real difficulties that comes with cracking an RSA cipher and a Diffie-Hellman Key Exchange is the factorization of large numbers, especially when these large numbers have large prime factors. However, there are differences in how each one is attacked and how each one can have a breach of security. The Diffie-Hellman Key Exchange can be solved by using the Discrete Logarithm Problem. The reason why this is beneficial is because using the Discrete Logarithm Problem allows for more randomness, and it makes the encryption less predictable, as displayed below:




The above illustration indicates the randomness of the discrete logarithm, when picking viable numbers $a$ and $n$ that satisfy the DLP conditions in our discrete logarithm formula of $n = log_g(a)$ [2]. There is no clear-cut trend among the data here, when picking 2 relatively prime numbers for our operation. Here is a data description of picking prime numbers $p$ and $q$ for our RSA operation:




The above is a graph showing the results of picking two prime numbers for RSA calculation. Notice the trend in the data [4].

We see that in the intersection of the red and blue regions of this graph, there is a clear trend. There is an uphill, a peak in the data and then there is a downhill. RSA on its own may be susceptible to hacker attacks by using the behavior of these prime numbers together. Although it may be stronger when using larger prime moduli, there are still behaviors among prime numbers in which onlookers can observe through modular arithmetic.

Although the RSA cipher is the more well known and more widely-used cipher, the Diffie-Hellman Key Exchange when coupled with discrete logarithms is also very effective. However, the Diffie-Hellman Key Exchange may also be susceptible to an attack known as "the man-in the middle" attacks(MITM). This is because there is no private domain in the Diffie-Hellman Key Exchange, so an onlooker could potentially break the cipher using a key pair of numbers. An example of such an attack would be the baby-step giant-step algorithm, in which we refer the reader to [3]

On the other hand, the RSA cipher has both domains, so there is information that is unknown to the public. However, there are problems and holes in the RSA cipher, one of which is its malleability. If we change the key in certain ways, the exponents will also change in a certain way, which will allow for someone to be able to find a connection between the old key and the new key. Recently this problem has been fixed with the use of digital signatures, which allows for a larger private domain in order to further strengthen the cipher. The reader can find more information on digital signatures here: [1].

In conclusion, the RSA cipher is a lot more secure than the Diffie-Hellman Key Exchange, however coupling the Diffie-Hellman Key Exchange with discrete logarithms is a promising part of our technological security that can be further looked at and implemented.

## References

[1] CISA, Understanding digital signatures, (2009).

[2] R.C Daileda, *Diffie-hellman key exchange and the discrete log problem*, (2020).

[3] William Gasarch, *Baby-step/giant step dl algorithm*, (2015).

[4] Kumari, *Implementation differences rsa key generation leads to weaker keys.*

[5] Douglas Robert Stinson and Maura Paterson, *Cryptography: theory and practice*, CRC press, (2018).