

**Cyber SECURITY Report project**



Crystle Yi

**Overview**

Alice and Bob are [fictional](https://en.wikipedia.org/wiki/Fiction) [characters](https://en.wikipedia.org/wiki/Character_(arts)) commonly used as a [placeholder name](https://en.wikipedia.org/wiki/Placeholder_name) in [cryptology](https://en.wikipedia.org/wiki/Cryptology), and in other science and engineering literature where there are several participants in a [thought experiment](https://en.wikipedia.org/wiki/Thought_experiment).

We’ll be using Alice and Bob for the Diffie-Hellman and the RSA algorithms as well as explaining in depth about these algorithms.

**The Diffie-Hellman**

The Diffie-Hellman key exchange was one of the most important developments in public-key cryptography and it is still frequently implemented in a range of today’s different security protocols.

It allows two parties who have not previously met to securely establish a key which they can use to secure their communications. In this article, we’ll explain what it’s used for, how it works on a step-by-step basis, its different variations, as well as the security considerations that need to be noted in order to implement it safely.

**What is the Diffie-Hellman key exchange?**

The Diffie-Hellman key exchange was the first widely used method of safely developing and exchanging keys over an insecure channel.

It may not seem so exciting or groundbreaking in the above terms, so let’s give an example that explains why the Diffie-Hellman key exchange was such an important milestone in the world of cryptography, and why it is still so frequently used today.

The most common solution would be to encrypt the message with a code. The easiest way is to prearrange whichever type of code and key you plan on using beforehand, or to do it over a safe communication channel.

Let’s say you want to communicate with a spy from an allied nation who you have never met before. You don’t have a secure channel over which to talk to them. If you don’t encrypt your message, then any adversary who intercepts it will be able to read the contents. If you encrypt it without telling the ally the code, then the enemy won’t be able to read it, but neither will the ally.

The Diffie-Hellman key exchange was the first publicly used mechanism for solving this problem. The algorithm allows those who have never met before to safely create a shared key, even over an insecure channel that adversaries may be monitoring.

**The history of the Diffie-Hellman key exchange**

The Diffie-Hellman key exchange traces its roots back to the 1970s. While the field of cryptography had developed significantly throughout the earlier twentieth century, these advancements were mainly focused in the area of symmetric-key cryptography.

It wasn’t until 1976 that public-key algorithms emerged in the public sphere, when Whitfield Diffie and Martin Hellman published their[paper](https://ee.stanford.edu/~hellman/publications/24.pdf), New Directions in Cryptography. The collaboration outlined the mechanisms behind a new system, which would come to be known as the Diffie-Hellman key exchange.

The work was partly inspired by earlier developments made by Ralph Merkle. The so-called[Merkle’s Puzzles](http://www.merkle.com/1974/PuzzlesAsPublished.pdf) involve one party creating and sending a number of cryptographic puzzles to the other. These puzzles would take a moderate amount of computational resources to solve.

**Where is the Diffie-Hellman key exchange used?**

The main purpose of the Diffie-Hellman key exchange is to securely develop shared secrets that can be used to derive keys. These keys can then be used with symmetric-key algorithms to transmit information in a protected manner. Symmetric algorithms tend to be used to encrypt the bulk of the data because they are more efficient than public key algorithms.

Technically, the Diffie-Hellman key exchange can be used to establish public and private keys. However, in practice, RSA tends to be used instead. This is because the RSA algorithm is also capable of signing public-key certificates, while the Diffie-Hellman key exchange is not.

The ElGamal algorithm, which was used heavily in PGP, is based on the Diffie-Hellman key exchange, so any protocol that uses it is effectively implementing a kind of Diffie-Hellman.

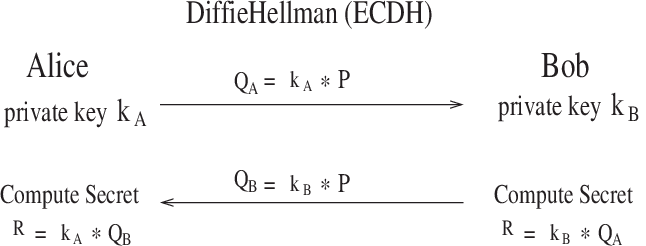
As one of the most common methods for safely distributing keys, the Diffie-Hellman key exchange is frequently implemented in security protocols such as TLS, IPsec, SSH, PGP, and many others. This makes it an integral part of our secure communications.

As part of these protocols, the Diffie-Hellman key exchange is often used to help secure your connection to a website, to remotely access another computer, and for sending encrypted emails

**How does the Diffie-Hellman key exchange work?**

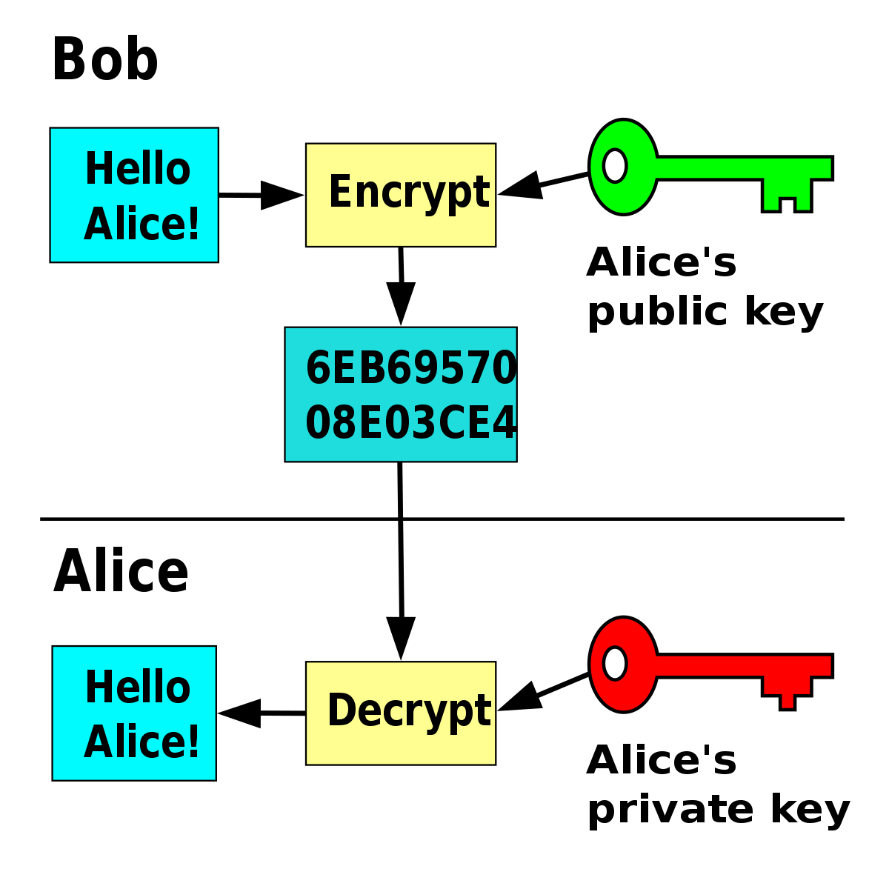
The Diffie-Hellman key exchange is complex, and it can be difficult to get your head around how it works. It uses very large numbers and a lot of math, something that many of us still dread from those long and boring high school lessons.

To make things a bit easier to understand, we will start by explaining the Diffie-Hellman key exchange with an analogy. Once you have a big-picture idea of how it works, we’ll move on to a more technical description of the underlying processes.



**Diffie–Hellman Key Agreement**

The Diffie Hellman key agreement protocol establishes a session key without using any redistributed keys. The messages exchanged between Alice and Bob can be read by anyone able to eavesdrop, and yet the eavesdropper won't know the session key that Alice and Bob end up with. On the other hand, Diffie–Hellman doesn't authenticate the participants. Since it is rarely useful to communicate securely without being sure whom you're communicating with, Diffie Hellman is usually augmented in some way to provide authentication. One of the main uses of Diffie–Hellman is in the Internet Key Exchange (IKE) protocol, a central part of the IP Security (IPsec) architecture.



**How to encrypt a big file using OpenSSL and someone's public key:**

* Step 0) Get their public key. The other person needs to send you their public key in .pem format.
* Step 1) Generate a 256 bit (32 byte) random key. openssl rand -base64 32 > key.bin.
* Step 2) Encrypt the key.
* Step 3) Actually Encrypt our large file.
* Step 4) Send/Decrypt the files

**UDP Connection:**

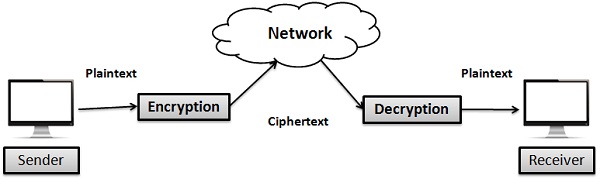
User datagram protocol (UDP) operates on top of the Internet Protocol (IP) to transmit datagrams over a network. UDP does not require the source and destination to establish a three-way handshake before transmission takes place. Additionally, there is no need for an end-to-end connection.

**Key Establishment:**

Key exchange (also key establishment) is a method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. If the sender and receiver wish to exchange encrypted messages, each must be equipped to encrypt messages to be sent and decrypt messages received.

**Data Encryption and Decryption:**

Encryption is the process of translating plain text data (plaintext) into something that appears to be random and meaningless (ciphertext). Decryption is the process of converting ciphertext back to plaintext. A symmetric key is used during both the encryption and decryption processes.



**Proper and clear message Display:**

* Cout is used to Print Message on the Screen.
* Insertion Operator is Used to put data on screen (<<)
* Return type of main function is integer as it returns 0 to operating system upon successful execution of program.

**The Diffie-Hellman Key Exchange**

[Diffie-Hellman key exchange](https://searchsecurity.techtarget.com/definition/Diffie-Hellman-key-exchange), also called an exponential key exchange, is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of an intended code breaker mathematically overwhelming. Diffie–Hellman key exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network and uses public-key techniques to allow the exchange of a private encryption key.

In order to simplify the explanation of how the algorithm works, we will use small positive integers. In reality, the algorithm uses large numbers. In addition, you may find fairly easy explanations on [Wikipedia](https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange) and [Khan Academy](https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/diffie-hellman-key-exchange-part-2).

Communicating in the clear, Alice and Bob agree on two positive integers, a prime number, and a generator. A [generator](https://searchsecurity.techtarget.com/definition/Diffie-Hellman-key-exchange) is a number that, when raised to positive whole-number powers less than the prime number, never produces the same result for any two such whole numbers. Let us assume that Alice will use the prime number 17 and Bob the generator 3. Then Alice selects a private random number, say 15, and calculates 315mod17 which equals 6 and sends the result publicly to Bob.  
  
Then Bob selects his private random number, say 13, calculates 313mod17 and sends the result (which is 12) publicly to Alice. The heart of the trick is the following computation. Alice takes Bob’s public result (=12) and calculates 1215mod17. The result (=10) is their shared secret key. On the other hand, Bob takes Alice’s public result (=6) and calculates 613mod17 which results again to the same shared secret. Now Alice and Bob can communicate using the symmetric algorithm of their choice and the shared secret key, which was never transmitted over the insecure circuit.

If a third party was listening to the exchange, it would be computationally difficult for this party to determine the secret key. In fact, when using large numbers, this action is computationally expensive for modern supercomputers to do in a reasonable amount of time.

**RSA**

[RSA](https://searchsecurity.techtarget.com/definition/RSA) is a cryptosystem for public-key encryption and is widely used for securing sensitive data, particularly when being sent over an insecure network such as the Internet. RSA was first described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology. Public-key cryptography, also known as asymmetric cryptography, uses two different but mathematically linked keys, one public and one private. The public key can be shared with everyone, whereas the private key must be kept secret. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm: It provides a method of assuring the confidentiality, integrity, authenticity, and non-reputability of electronic communications and data storage.

RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total, that’s factoring, is considered infeasible due to the time it would take even using today’s super computers. The RSA algorithm [involves four steps](https://en.wikipedia.org/wiki/RSA_(cryptosystem)):

* key generation
* key distribution
* encryption
* decryption

The public and the private key-generation algorithm is the most complex part of RSA cryptography and falls beyond the scope of this post. You may find an example on [Tech Target](https://searchsecurity.techtarget.com/definition/RSA).

**RSA Algorithm**

RSA involves a public key and private key. The public key can be known to everyone- it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key.

RSA (Rivest–Shamir–Adleman) is a [public-key cryptosystem](https://en.wikipedia.org/wiki/Public-key_cryptography) that is widely used for secure data transmission. It is also one of the oldest. The [acronym](https://en.wikipedia.org/wiki/Acronym) RSA comes from the surnames of [Ron Rivest](https://en.wikipedia.org/wiki/Ron_Rivest), [Adi Shamir](https://en.wikipedia.org/wiki/Adi_Shamir), and [Leonard Adleman](https://en.wikipedia.org/wiki/Leonard_Adleman), who publicly described the algorithm in 1977. An equivalent system was developed secretly, in 1973 at [GCHQ](https://en.wikipedia.org/wiki/Government_Communications_Headquarters) (the British [signals intelligence](https://en.wikipedia.org/wiki/Signals_intelligence) agency), by the English mathematician [Clifford Cocks](https://en.wikipedia.org/wiki/Clifford_Cocks).

In a public-key [cryptosystem](https://en.wikipedia.org/wiki/Cryptosystem), the [encryption key](https://en.wikipedia.org/wiki/Encryption_key) is public and distinct from the [decryption key](https://en.wikipedia.org/wiki/Decryption_key), which is kept secret (private). An RSA user creates and publishes a public key based on two large [prime numbers](https://en.wikipedia.org/wiki/Prime_number), along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decoded by someone who knows the prime numbers.

The security of RSA relies on the practical difficulty of [factoring](https://en.wikipedia.org/wiki/Factorization) the product of two large [prime numbers](https://en.wikipedia.org/wiki/Prime_number), the "[factoring problem](https://en.wikipedia.org/wiki/Factoring_problem)". Breaking RSA [encryption](https://en.wikipedia.org/wiki/Encryption) is known as the [RSA problem](https://en.wikipedia.org/wiki/RSA_problem). Whether it is as difficult as the [factoring problem](https://en.wikipedia.org/wiki/Factoring_problem) is an open question. There are no published methods to defeat the system if a large enough key is used.

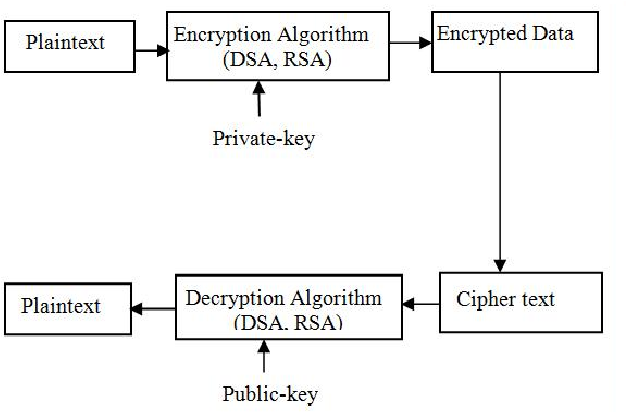
RSA is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, RSA is used to transmit shared keys for [symmetric key](https://en.wikipedia.org/wiki/Symmetric-key_algorithm) cryptography, which are then used for bulk encryption-decryption.

**Operation**

The RSA algorithm involves four steps: [key](https://en.wikipedia.org/wiki/Key_(cryptography)) generation, key distribution, encryption, and decryption.

A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d, and n.

and that knowing e and n, or even m, it can be extremely difficult to find d. The [triple bar](https://en.wikipedia.org/wiki/Triple_bar) (≡) here denotes [modular congruence](https://en.wikipedia.org/wiki/Modular_arithmetic).



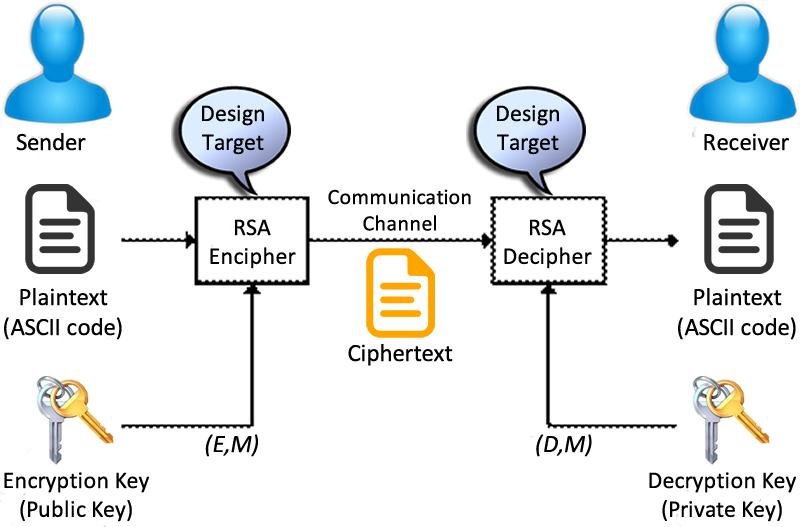
In addition, for some operations it is convenient that the order of the two exponentiations can be changed.

RSA involves a public key and a [private key](https://en.wikipedia.org/wiki/Private_key). The public key can be known by everyone, and it is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time by using the private key. The public key is represented by the integers n and e; and, the private key, by the integer d (although n is also used during the decryption process, so it might be considered to be a part of the private key, too). m represents the message

**Key generation**

The keys for the RSA algorithm are generated in the following way:

* Choose two distinct [prime numbers](https://en.wikipedia.org/wiki/Prime_number) p and q.
* For security purposes, the integers p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder. Prime integers can be efficiently found using a [primality test](https://en.wikipedia.org/wiki/Primality_test).
* The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d, which must be kept secret. p, q, and λ(n) must also be kept secret because they can be used to calculate d. In fact, they can all be discarded after d has been computed.



**What are the differences?**

Both RSA and Diffie-Hellman are public-key encryption algorithms [strong enough for commercial purposes](https://searchsecurity.techtarget.com/answer/Choosing-the-right-public-key-algorithm-RSA-vs-Diffie-Hellman) because they are both based on supposedly intractable problems, the difficulty of factoring large numbers and exponentiation and modular arithmetic respectively. The minimum recommended key length for encryption systems is 128 bits, and both exceed that with their 1,024-bit keys. Both have been [subjected to scrutiny](https://searchsecurity.techtarget.com/answer/Diffie-Hellman-vs-RSA-Comparing-key-exchange-algorithms) by mathematicians and cryptographers, but given correct implementation, neither is significantly less secure than the other.

The [nature](https://searchsecurity.techtarget.com/definition/Diffie-Hellman-key-exchange) of the Diffie-Hellman key exchange, however, makes it susceptible to man-in-the-middle (MITM) attacks, since it doesn't authenticate either party involved in the exchange. The MITM maneuver can also create a key pair and spoof messages between the two parties, who think they're both communicating with each other. This is why Diffie-Hellman is used in combination with an additional authentication method, generally digital signatures.

Unlike Diffie-Hellman, the RSA algorithm can be used for [signing digital signatures](https://searchsecurity.techtarget.com/answer/Diffie-Hellman-vs-RSA-Comparing-key-exchange-algorithms) as well as symmetric key exchange, but it does require the exchange of a public key beforehand. However, [recent research](https://eprint.iacr.org/2018/1173.pdf) has demonstrated that even 2048-bits long RSA keys can be effectively downgraded via either man-in-the-browser or padding oracle attacks. The [report](https://eprint.iacr.org/2018/1173.pdf) suggests that the safest countermeasure is to deprecate the RSA key exchange and switch to (Elliptic Curve) Diffie-Hellman key exchanges.

**Program Code of RSA Algorithm:**

/\*

\* C++ Program to Implement the RSA Algorithm

\*/

#include<iostream>

#include<math.h>

#include<string.h>

#include<stdlib.h>

using namespace std;

long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;

char msg[100];

int prime(long int);

void ce();

long int cd(long int);

void encrypt();

void decrypt();

int prime(long int pr)

{

int i;

j = sqrt(pr);

for (i = 2; i<= j; i++)

{

if (pr % i == 0)

return 0;

}

return 1;

}

int main()

{

cout<<"\nSecure Communication Channel\n";

cout<< "\nAlice Enter Secret Key For Communication\n";

cin>> p;

flag = prime(p);

if (flag == 0)

{

cout<< "\nWRONG INPUT\n";

exit(1);

}

cout<< "\nBob Enter Secret Key For Communication\n";

cin>> q;

cout<< "\nAliceAnd bob Exchange the Key successsfully\n";

flag = prime(q);

if (flag == 0 || p == q)

{

cout<< "\nWRONG INPUT\n";

exit(1);

}

cout<< "\nEnter the Message Which You Want To Send!!!\n";

fflush(stdin);

cin>> msg;

for (i = 0; msg[i] != '\0'; i++)

m[i] = msg[i];

n = p \* q;

t = (p - 1) \* (q - 1);

ce();

cout<< "\nPOSSIBLE VALUES OF e AND d ARE\n";

for (i = 0; i< j - 1; i++)

cout<< e[i] << "\t" << d[i] << "\n";

encrypt();

decrypt();

return 0;

}

void ce()

{

int k;

k = 0;

for (i = 2; i< t; i++)

{

if (t % i == 0)

continue;

flag = prime(i);

if (flag == 1 &&i != p &&i != q)

{

e[k] = i;

flag = cd(e[k]);

if (flag > 0)

{

d[k] = flag;

k++;

}

if (k == 99)

break;

}

}

}

long int cd(long int x)

{

long int k = 1;

while (1)

{

k = k + t;

if (k % x == 0)

return (k / x);

}

}

void encrypt()

{

long int pt, ct, key = e[0], k, len;

i = 0;

len = strlen(msg);

while (i != len)

{

pt = m[i];

pt = pt - 96;

k = 1;

for (j = 0; j < key; j++)

{

k = k \* pt;

k = k % n;

}

temp[i] = k;

ct = k + 96;

en[i] = ct;

i++;

}

en[i] = -1;

cout<< "\nTHE ENCRYPTED MESSAGE IS\n";

for (i = 0; en[i] != -1; i++)

printf("%c", en[i]);

}

void decrypt()

{

long int pt, ct, key = d[0], k;

i = 0;

while (en[i] != -1)

{

ct = temp[i];

k = 1;

for (j = 0; j < key; j++)

{

k = k \* ct;

k = k % n;

}

pt = k + 96;

m[i] = pt;

i++;

}

m[i] = -1;

cout<< "\nTHE DECRYPTED MESSAGE IS\n";

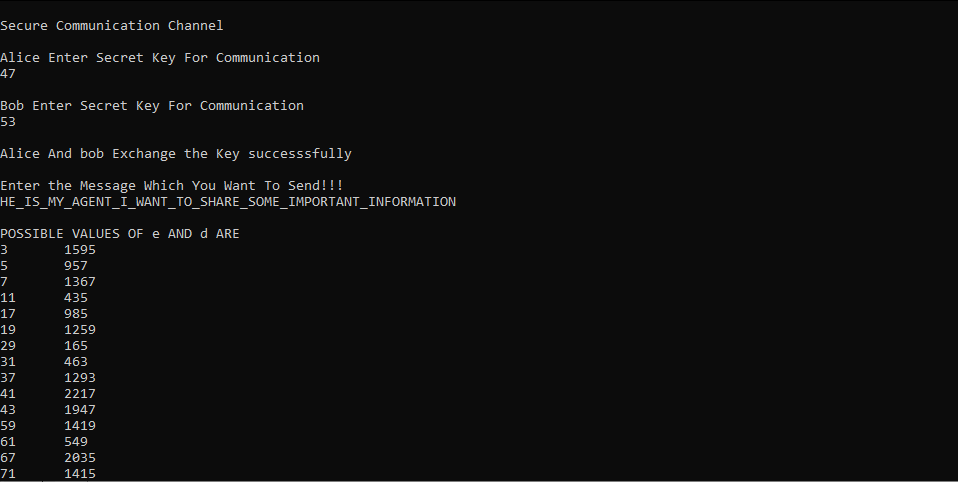
for (i = 0; m[i] != -1; i++)

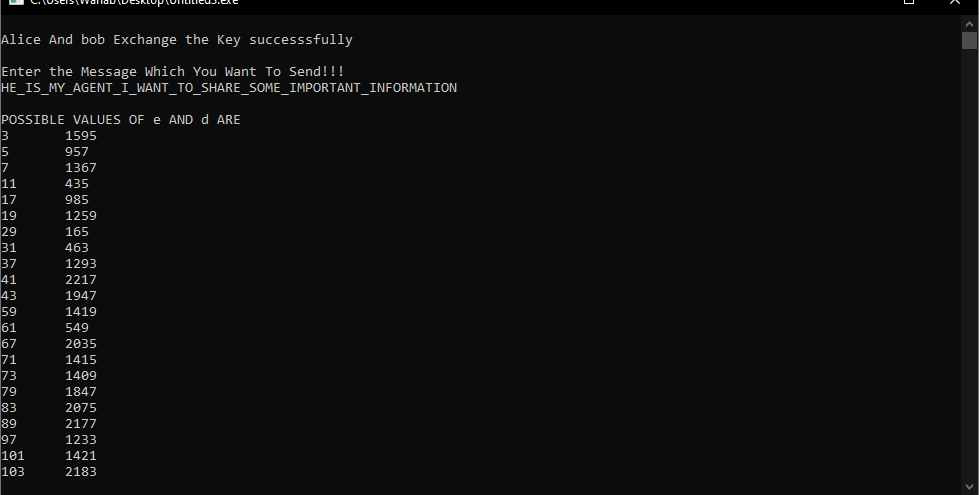
printf("%c", m[i]);

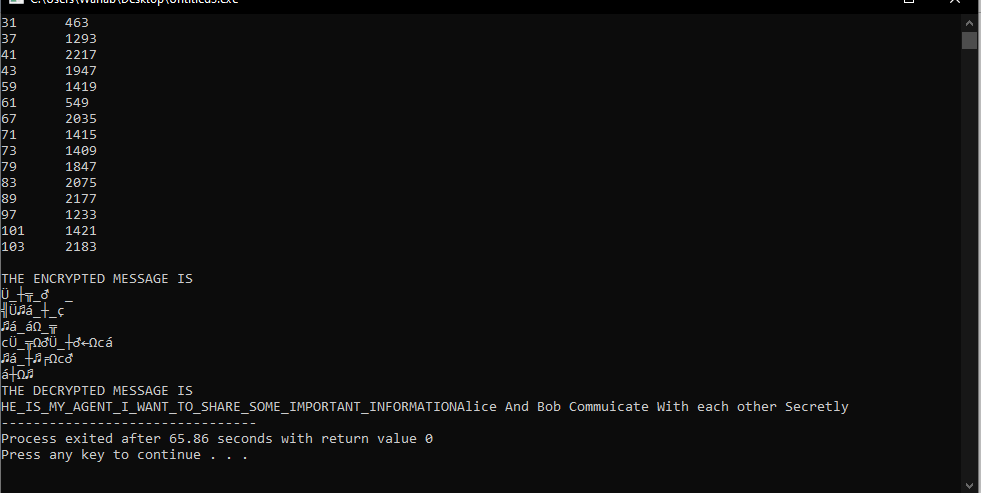
cout<<"Alice And Bob Commuicate With each other Secretly ";

}

**OUTPUT:**







**Program Code of Diffie Hellman Algorithm:**

#include<iostream>

#include<cstdio>

using namespace std;

class DiffieHellman

{

public:

long long int p,g,x,a,y,b,A,B;

DiffieHellman(long long int p1,long long int g1,long long int x1,long long int y1)

{

p = p1;

g = g1;

x = x1;

y = y1;

a=power(g,x,p);

b=power(g,y,p);

A = power(b,x,p);

B = power(a,y,p);

}

long long int power(int a,int b,int mod)

{

long long int t;

if(b==1)

return a;

t=power(a,b/2,mod);

if(b%2==0)

return (t\*t)%mod;

else

return (((t\*t)%mod)\*a)%mod;

}

};

int main()

{

long long int p,g,x,a,y,b,A,B;

cout<<" Diffie Hellman Alogorithm\n\n";

cout<<"Enter the values of p and g upon which Alice And Bob both will aggree : "<<endl;

cin>>p>>g;

cout<<"Enter the Secret Integer for Alice : ";

cin>>x;

cout<<"Enter the Secret Integer for Bob : ";

cin>>y;

cout<<endl;

DiffieHellman dh(p,g,x,y);

cout<<"Alice's private key, known only to Alice : "<<dh.a<<endl;

cout<<"Bob's private key known only to Bob : "<<dh.b<<endl;

cout<<endl;

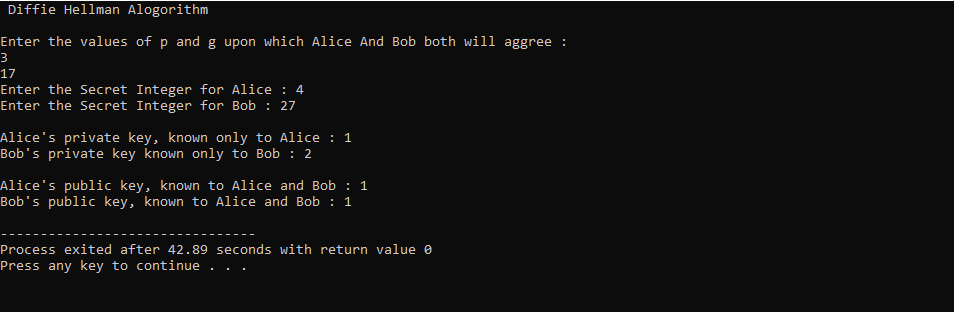
cout<<"Alice's public key, known to Alice and Bob : "<<dh.A<<endl;

cout<<"Bob's public key, known to Alice and Bob : "<<dh.B<<endl;

return 0;

}

**Output:**

****

**Reference:**

1. Rivest, R. L.; Shamir, A.; Adleman, L. (1978-02-01). "A Method for Obtaining Digital Signatures and Public-key Cryptosystems". Communications of the ACM. 21 (2): 120–126. [CiteSeerX](https://en.wikipedia.org/wiki/CiteSeerX_(identifier)) [10.1.1.607.2677](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.607.2677). [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1145/359340.359342](https://doi.org/10.1145%2F359340.359342). [ISSN](https://en.wikipedia.org/wiki/ISSN_(identifier)) [0001-0782](https://www.worldcat.org/issn/0001-0782).
2. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-2) Newton, David E. (1997). Encyclopedia of Cryptography. Santa Barbara California: Instructional Horizons, Inc. p. 10.
3. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-3) Rivest, R. L., A. Shamir, and L. Adleman. “On Digital Signatures and Public-Key Cryptosystems.” Cambridge MA: Massachusetts Institute of Technology, April 1977.
4. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-4) Rivest, R. L., A. Shamir, and L. Adleman. Cryptographic Communications System and Method. 4405829. Cambridge MA, filed December 14, 1977, and issued September 20, 1983.
5. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-5) Bob Brown (7 February 2005). ["Security's inseparable couple: Alice & Bob"](https://www.networkworld.com/article/2318241/lan-wan-security-s-inseparable-couple.html). NetworkWorld.
6. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-6) Rabin, Michael O. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.
7. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-7) Blum, Manuel. “Coin Flipping by Telephone a Protocol for Solving Impossible Problems.” ACM SIGACT News 15, no. 1 (November 10, 1981): 23–27.
8. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-8) Blum, Manuel (1983). "How to exchange (Secret) keys". ACM Transactions on Computer Systems. 1 (2): 175–193. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1145/357360.357368](https://doi.org/10.1145%2F357360.357368).
9. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-9) [doi](https://en.wikipedia.org/wiki/Doi_(identifier)" \o "Doi (identifier)):[10.1016/j.jvlc.2007.05.001](https://doi.org/10.1016%2Fj.jvlc.2007.05.001)
10. [^](https://en.wikipedia.org/wiki/Alice_and_Bob#cite_ref-10) Gordon, John. “The Alice and Bob After Dinner Speech.” Zurich, April 1984. <http://downlode.org/Etext/alicebob.html>.