Elliptic Curve Cryptography

Professors: Manish Gupta

Group Members and their Contribution

1. **Dev Changela(202101069)**
   Contribution:
2. **Swet Lakhani(202101218)**
   Contribution:
3. **Jay Sabva(202101224)**
   Contribution:
4. **Soham Viradiya(202101472)**
   Contribution:
5. **Vandit Bhalani(202101478)**
   Contribution:
6. **Shrut Kalathiya(202101479)**
   Contribution:
7. **Vasu Golakiya(202101487)**
   Contribution:

1 Introduction and History of Cryptography

The History of cryptography can be split into two eras:the classical era and the modern era. The turning point between the two occurred in 1977, when both the RSA algorithm and the Diffie-Hellman key exchange algorithms were introduced.These new algorithms were revolutionary because they represented the first viable cryptographic schemes where security was based on the theory of numbers; it was the first to enable secure communication between two parties without a shared secret.Cryptography went from being about securely transporting secret codebooks around the world to being able to have provably secure communication between any two parties without worrying about someone listening in on the key exchange.

Modern cryptography is founded on the idea that the key that you use to encrypt your data can be made public while the key that is used to to decrypt your data can be kept private. As such, these systems are known as public key cryptographic systems. The first, and still most widely used of these systems, is known as RSA — named after the initials of the three men who first publicly described the algorithm: Ron Rivest, Adi Shamir and Leonard Adleman.

What you need for a public key cryptographic system to work is a set of algorithms that is easy to process in one direction, but difficult to undo. In the case of RSA, the easy algorithm multiplies two prime numbers. If multiplication is the easy algorithm, its difficult pair algorithm is factoring the product of the multiplication into its two component primes. Algorithms that have this characteristic — easy in one direction, hard the other — are known as Trap door Functions. Finding a good Trapdoor Function is critical to making a secure public key cryptographic system. Simplistically: the bigger the spread between the difficulty of going one direction in a Trapdoor Function and going the other, the more secure a cryptographic system based on it will be.

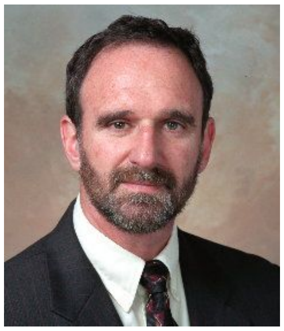
Figure 1: Martin Hellman

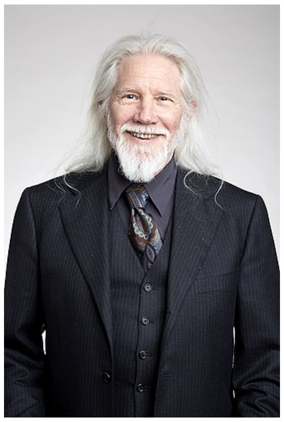
Figure 2: Whitefield Diffle

2 Diffie-Hellman Key Exchange Algorithm(DH-key X change Algorithm)

2.1 Introduction

Whitefield and Martin Hellman develop Diffie-Hellman key exchange Algorithms in 1976 to overcome the problem of key agreement and exchange. It was one of the first public-key exchange protocols as conceived by Ralph Merkle and named after Whitefield Diffe and Martin Hellman. It enables the two parties who want to communicate with each other to agree on a symmetric key(a key that can only be used for encrypting and decrypting).

This Algorithm can only be used for Key Exchange and not for encryption and decryption process.The Algorithm is based on mathematical principles.

2.2 Reason to use Algorithm

When we are sending a key to receiver then it can be attacked in between(i.e. Insecurity may occurs). It allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel.

2.3 Algorithm

1. Either Alice or Bob selects a large, secure prime p and a primitive root α(mod p). Both p and α can be made public.
2. Alice chooses a secret random $x(1 <= x <= p-2)$ and Bob selects a secret random key $y(1 <= y <= p-2)$.
3. Alice sends α^x(mod p) to Bob and Bob sends α^y(mod p) to Alice.
4. Using this messages they receive, they can calculate the session key.

For Alice, k=$(\alpha^y)^x$(mod p)

For Bob, k=$(\alpha^x)^y$(mod p)

$\because$ Both have same number k, a key could be, middle 56 bits of k to obtain a DES key.

$\therefore$ System is secure as discrete log is difficult to solve.

Step-by-Step Explanation of above Algorithm:-

Alice	Bob
Public Keys available = p, α	Public Keys available = p, α
Selects Random private key = x	Selects Random private key = y
Key generated = a = α^x(mod p)	Key generated = b = α^y(mod p)
Exchange of generated keys takes place	Exchange of generated keys takes place
Key received = b	Key received = a
Generated Secret Key = ka = b^x(mod p)	Generated Secret Key = kb = a^y(mod p)

Algebrically, ka = kb. So, Now users have same secret key for communicating.

2.4 Example of Algorithm

1. Alice, Bob and Eve are three persons.Now, Alice and Bob selects two public keys p=23 and α= 9 which is known to Eve as well.
2. Alice selects a private key x= 4 and Bob selects a private key y= 3 which is not known to Eve.
3. Alice and Bob calculates public values a = 9^4(mod 23) = 6 and b= 9^3(mod 23) = 16 respectively which is also not known to Eve.
4. Alice and Bob exchanges a and b. So, by exchanging a and b, there is a public network by which Eve can know what is a and b.
5. Alice receives public key b=16 and Bob receives public key a=6.
6. Alice and Bob computes ka = 16^4(mod 23) = 9 and kb = 6^3(mod 23) = 9 respectively which can't be determined by Eve either.
7. 9 is the secret key. As this is not determined by Eve, So this remains a private key.

Alice	
Known	Unknown
p = 23	
α = 9	
x = 4	y
a = 9^4(mod 23) = 6	
ka=16^4(mod 23) = 9	

Bob	
Known	Unknown
p = 23	
α = 9	
y = 3	x
b= 9^3(mod 23) = 9	
kb=6^3(mod 23) = 9	

Eve	
Known	Unknown
p = 23	
α = 9	
	x,y
a,b	
	ka,kb

2.5 Practical Example of Algorithm:-

To understand above Algorithm we need to take one Analogical example of colors i.e. The process begins by having the two person, Alice and Bob, publicly agree on any random starting color that does not need to be keep secret(i.e. it's public). In this example, the color is yellow. Each person also selects a secret color that they keep to themselves – in this case, red and cyan. The main part of the process is that Alice and Bob each mix their own secret color together with their mutually shared color, resulting in orange-tan and light-blue mixtures respectively, and then publicly exchange the two mixed colors. Finally, each of them mixes the color they received from the partner with their own private color. The result is a final color mixture (yellow-brown in this case) that is identical to their partner's final color mixture.

If a third person listened to the exchange, it would only know the common color (yellow) and the first mixed colors (orange-tan and light-blue), but it would be difficult for this person to determine the final secret color (yellow-brown). Bringing the analogy back to a real-life exchange using large numbers rather than colors, this determination is computationally expensive. It is impossible to compute in a practical amount of time even for modern supercomputers.

2.6 Advantages

Many Protocols uses this Algorithm for enhancing their securities which includes:-

1. Secure Shell(SSH)
2. Transport Layer Security (TLS) / Secure Sockets Layer (SSL)
3. Public Key Infrastructure (PKI)
4. Internet Key Exchange (IKE)
5. Internet Protocol Security (IPSec)

2.7 Limitations

1. Lack of authentication procedure
2. Algorithm can only be used for exchanging symmetric key.
3. As there is no authentication involved, it is vulnerable to man-in-the-middle attack.
4. As it is computationally intensive, it is expensive in terms of resources and CPU performance time.
5. Encryption of information cannot be performed with the help of this algorithm
6. Digital Signature cannot be signed using this algorithm.

2.8 Code

```
#include <iostream>
#include<math.h>
using namespace std;

long long int power(long long int P, long long int a, long long int G){
    long long int ans = 1;
    while(a--){
        ans = (ans*P)%G;
    }
    return ans;
}

int main()
{
    long long int P,G,x,a,y,b,ka,kb;

    cout<<"Enter Largest prime Number:";
    cin>>G;

    cout<<"Enter the primitive root Number:";
    cin>>P;

    cout<<"Enter Private Key of ALice:";
    cin>>a;

    x=power(P,a,G);

    cout<<"Enter Private Key of Bob:";
    cin>>b;

    y=power(P,b,G);

    ka=power(y,a,G);
    kb=power(x,b,G);

    cout<<"Value of P(primitive root):"<<P<<endl;
    cout<<"Value of G(Largest prime Number):"<<G<<endl;

    cout<<"Private key for ALice is "<<a<<endl;
    cout<<"Private key for Bob is "<<b<<endl;

    cout<<"Secret key for ALice is "<<ka<<endl;
    cout<<"Secret key for Bob is "<<kb<<endl;

    return 0;
}
```