Key exchanging technique toward large static set for secure communication

C.SUSHAMA
Assistant Professor
Department of CSSE
Sree Vidyanikethan Engineering College, Tirupati

P.NEELIMA
Assistant Professor
Dept of CSE,
Siddhartha Educational Academy Group of Institutions,
C.Golapalli, Tirupati

M. SUNIL KUMAR
Research Scholar, Dept of CSE
S V University
Tirupati
.

Abstract— In this work we are going to implement two extension algorithms of basic Diffi-Hellman technique. Basically the Diffi-Hellman (DH) technique is used for exchanging the secret key between the two participants. This key is helpful for private key encryption. But it is not safe to exchange a secret key through public communication channels. Infarct the ability to dynamically and public establish a session key for secured communication is a big challenge in cryptography. The first extension is the multiple two-party keys algorithm. In this we are exchanging multiple keys between two parties instead of one key. If the two participants exchange two public keys then they can generate totally 15 shared keys. In which 4 keys are called base keys and remaining 11 keys are called as extended keys. The main advantages are decrease in the key exchange overhead and increase if additional protection to the keys and widening of applicability. The second algorithm is for the generation of one multi-party key. In this the key exchange technique is applied for a large static group. One among the group member acts as a group controller and key exchange is performed between the controller and the remaining group members. In addition to the key exchange, encryption and decryption of data is performed by making use of the keys thus generated. DES algorithm is used for performing encryption and decryption of data.

INTRODUCTION

1.1 CRYPTOGRAPHY:

Cryptographic systems are generically classified along three independent dimensions:

- **The type of operations used for transforming plaintext to cipher text.** All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped in to another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost(that is, that all operations be reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.

- **The number of keys used.** If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver each use a different key, the system is referred to as asymmetric, two-key, or public-key encryption.

- **The way in which the plaintext is processed.** A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

1.2 CONVENTIONAL CRYPTOGRAPHY:

Cryptography is the study of mathematical systems for solving two kinds of security problems: privacy and authentication. A privacy system prevents the extraction of information by unauthorized parties from messages transmitted over a public channel, thus assuring the sender of a message that it is being read only by the intended recipient. An authentication system prevents the unauthorized injection of messages into a public channel, assuring the receiver of a message of the legitimacy of its sender. A channel is considered public if its security is inadequate for the needs of its users. A channel such as a telephone line may therefore be considered private by some users and public by others. Any channel may be threatened with eavesdropping or injection or both, depending on its use. In telephone communication, the threat of injection is paramount, since the called party cannot determine which phone is calling. Eavesdropping, which requires the use of a wiretap, is technically more difficult and legally hazardous. In radio, by comparison, the situation is reversed. Eavesdropping is passive and involves no legal hazard, while injection exposes the illegitimate transmitter to discovery and prosecution. Having divided our problems into those of privacy and authentication we will sometimes further subdivide authentication into message authentication, which is the problem defined above, and user authentication, in which the only task of the system is to verify that an individual is who he claims to be. For example, the identity of an individual who presents a credit card must be verified, but there is no message which he wishes to transmit. In spite of this apparent

absence of a message in user authentication, the two problems are largely equivalent. In user authentication, there is an implicit message "I AM USER X," while message authentication is just verification of the identity of the party sending the message. Differences in the threat environments and other aspects of these two sub problems, however, sometimes make it convenient to distinguish between them.

2. PUBLIC KEY CRYPTOGRAPHY:

This is one category of the cryptographic systems. In this category there are two types of keys available for each and every participant in the communication network, They are a Public Key and a Private Key. The Public Key of a participant is known to every other participant in the network through which they communicate with each other. And the private key is single and kept secret for each participant. If a participant needs to communicate with an other participant by sending data to it in the encryption format then it encrypts the data by using the Public Key of the receiver and the appropriate algorithm for encryption. The receiver receives the data and decrypts it by using the Private Key of itself and it makes use of the same algorithm for decryption which was used for encryption.

2.1 PRINCIPLES:

Of equal importance to conventional encryption is public-key encryption, which finds use in message authentication and key distribution. This section looks first at the basic concept of public-key encryption and takes a preliminary look at key distribution issues.

Public-key encryption, first publicly proposed by Diffie and Hellman in 1976, is the first truly revolutionary advance in encryption in literally thousands of years. Public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns, such as are used in symmetric encryption algorithms. More important, public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to the symmetric conventional encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

Before proceeding, we should first mention several common misconceptions concerning public-key encryption. One is that public-key encryption is more secure from cryptanalysis than conventional encryption. In fact, the security of any encryption scheme depends on (1) the length of the key and (2) the computational work involved in breaking a cipher. There is nothing in principle about either conventional or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis. A second misconception is that public-key encryption is a general purpose technique that has made conventional encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that conventional encryption will be abandoned . Finally there is a feleing that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for conventional encryption. In fact some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler or any more efficient than those required for conventional encryption.

A public key encryption scheme has six ingredients

- **Plaintext:** This is the readable message or data that is fed in to the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and Private key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts.
- **Decryption algorithm:** This algorithm accepts the cipher text and the matching key and produces the original plaintext. Some of the Public Key Cryptographic schemes

2.2 DIFFIE-HELLMAN KEY EXCHANGE:

The first published pubic-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key Exchange. A number of commercial products employ this key exchange technique. The purpose of this algorithm is to enable two users to exchange a secret key securely that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of the keys.

This key exchange technique makes use of a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key Xa, calculate Ya, and send that to user B. User B responds by generating a private value Xb, calculating Yb to user A. Both users can now calculate the key. The necessary public values p and q would need to be known ahead of time. Alternatively, user A could pick values for p and q and include those in the first message.

As an example of another use of the Diffie-Hellman algorithm, suppose that a group of users each generate a long-lasting private value Xa and calculate a public value Ya. These public values, together with global public values for p and q, are stored in some central directory. At any time user B, can access user A's public value, calculate a secret key, and use that to send an encrypted message to user A. If the central directory is trusted, then this form of communication provides

both confidentiality and a degree of authentication. Because only A and B can determine the key, no other user can read the message using this key. However, the technique does not protect against replay attacks.

2.3 DIGITAL SIGNATURE STANDARD:

The National Institute of Standards and Technology (NIST) have published Federal Information Processing Standard FIPS PUB 186, known as the Digital Signature Standard (DSS). The DSS makes use of the SHA-1 and presents a new digital signature technique, the Digital Signature Algorithm (DSA). The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996; The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange.

2.4 ELLIPTIC-CURVE CRYPTOGRAPHY:

The proposed elliptic curve cryptosystems are analogs of existing schemes. It is possible to define elliptic curve analogs of the RSA cryptosystem and it is possible to define analogs of public-key cryptosystems that are based on the discrete logarithm problem (such as ElGamal encryption and the DSA for instance). The case of analogs to the discrete logarithm problem can be divided into two classes. In the first class the finite field is said to have odd characteristic (typically a large prime number) and in the second class the field is said to have characteristic 2. While at first sight this might be viewed as a somewhat technical distinction the choice of underlying field can have implications for both the security and the performance of the cryptosystem. This distinction is similar to one that is made between cryptosystems based on the discrete logarithm problem.

It is interesting to note that the problems of integer factorization and of discrete logarithms over a prime field appear to be of roughly the same difficulty. Techniques used to solve one problem can be adapted to tackle the other. As we have mentioned, there are elliptic curve analogs to RSA but it turns out that these are chiefly of academic interest since they offer essentially no practical advantages over RSA. This is primarily the case because elliptic curve variants of RSA actually rely for their security on the same underlying problem as RSA, namely that of integer factorization.

The situation is different with variants of discrete logarithm cryptosystems. The security of the elliptic curve variants of discrete logarithm cryptosystems depends on a restatement of the conventional discrete logarithm problem for elliptic curves. This restatement is such that current algorithms that solve the conventional discrete logarithm problem in what is termed sub-exponential time are of little value in attacking the analogous elliptic curve problem. Instead the only available algorithms for solving these elliptic curve problems are more general techniques that run in what is termed exponential time.

The distinction between exponential and sub-exponential time for solving some problem is a vitally important one. In essence it means that methods of finding a solution to one problem are becoming infeasible much faster than those for solving the other problem. As we will see, this has considerable practical significance.

In this note we will only be considering elliptic curve cryptosystems that depend for their security on the problem of taking elliptic curve discrete logarithms. In particular we will be considering analogs to the DSA and to the ElGamal encryption scheme. These will be described as the Elliptic Curve DSA (ECDSA) and the Elliptic Curve Encryption Scheme (ECES) respectively.

At a high level, we can already make one important statement. On a functional level, mechanisms for encryption or digital signatures can be devised so that they depend on any of the three types of problems we have already mentioned; integer factorization, conventional discrete logarithms, and elliptic curve discrete logarithms. There are however many trade-offs between the systems and these depend on many circumstances. It is the purpose of this note to give some guidance as to the implications of these potential differences.

2.5 PRIVATE KEY CRYPTOGRAPHY:

This is the another category of the cryptographic systems. In this category there is only one key available with each and every participant in the communication network. That key is called as the Private Key. This private is kept secret and is not known to anybody in the network. If a participant needs to communicate with an other participant by sending data to it in the encryption format then it encrypts the data by using the Private Key of its own and the appropriate algorithm for encryption. The receiver receives the data and decrypts it by using the Private key of itself and it makes use of the same algorithm for decryption which was used for encryption.

Some of the Private Key Cryptographic schemes are

2.5.1 Data Encryption Standard (DES):

DES algorithm is used for both encryption and decryption of data.The encryption and decryption can be done under provision of a key. The key must be an 8 digit integer. DES encrypts and decrypts data in 64-bit blocks, using a 64-bit key (although the effective key strength is only 56 bits, as explained below). It takes a 64-bit block of plaintext as input and outputs a 64-bit block of cipher text. Since it always operates on blocks of equal size and it uses both permutations and substitutions in the algorithm, DES is both a block cipher and a product cipher. DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the cipher text. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a key using a brute-force attack. So as the number of rounds increases, the security of the algorithm increases exponentially.

3 PROPOSED METHODS:

The proposed methods overcome the demerits of the existing solutions. In this system we are introducing two extension algorithms of the basic Diffie-Hellman technique that is being used in the existing system. They are

1. Generation of multiple two-party keys
2. Generation of multi-party key

The description of these algorithms is given below.

3.1 GENERATION OF MULTIPLE TWO-PARTY KEYS:

In this extension instead of one key, multiple keys are shared between the participants. The number of keys that are shared depends upon the number of private keys that are assumed by each participant. If the two participants agree for two private keys then 15 keys are shared among them. The mathematical formula that gives the number of keys that are being shared between the two participants is nc1+nc2+nc3+…… ncn, where n stands for number of private keys that are assumed. In the case of two private keys 15 shared keys are generated. In these 15 keys 4 keys are called base keys. These are called so because these are the keys on the base of which the remaining 11 keys are generated. These keys are called extended keys because these keys are generated by multiplying the different combinations of the base keys. These keys are exchanged at a time and so the KEO overhead is reduced. Session communication can be established with the help of these keys available. So this is more secure than the existing system. This shared key is used for further communication like encryption and decryption of data using the DES algorithm.

3.2 GENERATION OF MULTI-PARTY KEY:

This extension is applicable for static group. Static group means a group of members in which the number of participants is fixed and they communicate among them. The existing system is not applicable for group communication, so it is extended for a static group. In a static group the number of participants is fixed. There will be one group controller who controls the overall performance of the group and remaining members of the group are called group members. They participate in the group communication as usual. A single shared key is exchanged between the controller and the remaining group members. The group controller and the group members contribute to share a single key. This shared key is used for further communication like encryption and decryption of data using the DES algorithm.

3.3 MERITS OF THE PROPOSED METHODS:

1. KEO overhead is reduced because more number of keys are shared at a time. Session communication may get rid of man-in-middle attack.
2. Because of the generation of multiple keys each can be used for each session.
3. Due to multiple keys generation communication through the public channels is more reliable.
4. Secure from Brute force attack and can be implemented in static groups.

4.ALGORITHM ANALYSIS:

Implementation is the stage in the paper where the theoretical design is turned into a working system and is giving confidence on the new system for the users, which it will work efficiently and effectively. It involves careful planning, investigation of the current System and its constraints on implementation, design of methods to achieve the change over, an evaluation, of change over methods. Apart from planning major task of preparing the implementation are education and training of users. The more complex system being implemented, the more involved will be the system analysis and the design effort required just for implementation.

4.1 PRELIMINARIES OF TWO-PARTY DH TECHNIQUE:

Diffie and Hellman introduced the concept of a two-party key-exchanging technique that allows two participants to exchange two public keys through unsecured channel and generate a shared secured key between them. Each party then combines the other's public key along with its own private key to generate a shared key. For clarity, a brief discussion of the DH technique follows.

Let A and B be the two participants that do not know anything about each other, but wish to establish a secured shared key between them. It consists of five steps as mentioned below.

Step 1: Both A and B agree on two large positive integers, n and g such that n is a prime number (and the order of a finite cyclic group) and g is a group generator.

Step 2: Party A randomly chooses a positive integer, x, which is smaller than n and serves as A's private key. Similarly, B chooses its own private key, y.

Step 3: Both A and B compute their public keys using X = g pow x mod n and Y = g pow y mod n, respectively.

Step 4: They exchange their public keys through a public communication channel.

Step 5: On receiving, both A and B compute their shared key K, using K = Y pow x mod n = g pow xy mod n and K = X pow y mod n = g pow xy mod n

4.1.1 GENERATION OF MULTIPLE TWO-PARTY KEYS:

The participant A generates two public keys as given below and sends to B X1 = g pow x1 mod n, X2 = g pow x2 mod n

where Xi and xi, for i = 1, 2, are the public and private keys of A, respectively. Similarly, the participant B chooses two private keys y1 and y2 randomly and generates two public keys as

Y1 = g pow y1 mod n, Y2 = g pow y2 mod n
where Y1 and Y2 are public keys.

These keys are transmitted to A. Now on exchange of two pairs of public keys, the participants can generate more than one shared key. Because instead of a single combination of the private keys (xy) as exist in basic DH, four combinations such as x1y1, x1y2, x2y1, x2y2 exist, and each of them can generate a DH-style key. The generation of four shared keys is given below.

The party A can compute the following keys

k1 = Y1 pow x1 mod n = g pow x1y1 mod n
k2 = Y2 pow x1 mod n = g pow x1y2 mod n
k3 = Y1 pow x2 mod n = g pow x2y1 mod n
k4 = Y2 pow x2 mod n = g pow x2y2 mod n

Similarly, the party B can compute the following keys

k1' = X1 pow y1 mod n = g pow x1y1 mod n
k2' = X1 pow y2 mod n = g pow x1y2 mod n
k3' = X2 pow y1 mod n = g pow x2y1 mod n
k4' = X2 pow y2 mod n = g pow x2y2 mod n

Since k1 = k1', k2 = k2', k3 = k3', and k4 = k4', thus four shared keys are generated. These four keys are termed as the base keys, because 11 additional shared keys can be derived by multiplying the base keys in different combinations. These keys are called extended keys as derived below.

If any two base keys, say k1 and k2 for instance, are multiplied, then we have

k5 = k1* k2 = g pow(x1y1+x1y2) mod n

Although the key k5 can be generated by the participants simply by multiplication, it is not true for the opponents. Because it appears to the opponents as a DDH problem such as

k5 = (Y1Y2) pow x1 mod n = (gy1 * gy2) pow x1 mod n =g pow (y1+y2)x1 mod n = g pow(x1y1+x1y2) mod n

or

k5 = (X1) pow (y1+y2) mod n = g pow (x1y1+x1y2) mod n

The key k5 is a shared key as each party individually can generate it. Also the private values xi and yi for i = 1, 2 are hidden; it is not possible for the opponents to determine k5 by knowing Y1, Y2, X1, g and n. Therefore this key appears to be secured (security analysis is given later).

Similarly, any two base keys can be considered for multiplication and note that out of four base keys there exists six (4C2) such multiplications, and therefore six shared keys are generated. The generation of the five shared keys is given below, where their expressions in terms of the public keys are given in the square brackets

k6 = k1 * k3 = g pow(x1y1+x2y1) mod n [(Y1) pow(x1+x2) mod n, (X1X2) pow y1 mod n]

k7 = k1 * k4 = g pow(x1y1+x2y2) mod n [(Y1 pow x1 * Y2 pow x2) mod n, (X1 pow y1 * X2 pow y2) mod n

k8 = k2 * k3 = g pow(x1y2+x2y1) mod n [(Y2 pow x1* Y1 pow x2) modn, (X1 pow y2 * X2 pow y1) mod n]

k9 = k2* k4 = g pow(x1y2+x2y2) mod n [(Y2) pow(x1+x2) mod n, (X1X2) pow y2 mod n]

k10 = k3 * k4 = g pow(x2y1+x2y2) mod n [(Y1Y2) pow x2 mod n, (X2) pow(y1+y2) mod n]

Now instead of two, if three base keys at a time are considered for multiplication, then we can have four such multiplications (4C3), and therefore another four shared keys are generated. The generation of these keys is given below

k11 = k1 * k2 * k3 = g pow(x1y1+x1y2+x2y1) mod n [(Y1 * Y2) pow x1 * Y1 pow x2 mod n, (X1 pow(y1+y2 * X2 pow y1) mod n]

k12 = k1 * k2 * k4 = g pow(x1y1+x1y2+x2y2) mod n [(Y1 pow x1 * Y2 pow(x1+x2))mod n, X1 pow(y1+y2) * X2 pow y2 mod n]

k13 = k1 * k3 * k4 = g pow(x1y1+x2y1+x2y2) mod n [(Y1) pow(x1+x2) * Y2 pow x2 mod n, X1 pow y1 * X2 pow (y1+y2)mod n]

k14 = k2 * k3 * k4 = g pow(x1y2+x2y1+x2y2) mod n [(Y2 pow(x1+x2) * Y1 pow x2) mod n, X1 pow y2 * X2 pow(y1+y2) mod n]

Finally, if all four base keys are multiplied together, a single shared key (4C4) is generated. The generation of the key is done using

k15 = (k1 * k2 * k3 * k4)modn = g pow(x1y1+ x1y2+ x2y1 +x2y2)mod n
[(Y1 * Y2)pow(x1+x2) mod n, (X1 * X2)(y1+y2) mod n

4.1.2 Generation of multi-party key:

We propose a contributory group key agreement protocol to exchange a multi-party key among the group members. In Group-DH, an arbitrary group member acts as a group controller that actually establishes the group-key and after which it becomes a normal member of the group. Let the group controller be Pi, where 1 _ i _ n for n-party group. Initially it itself forms a two-party group with each of the remaining group members, and produces (n 2 1) two-party groups. The Pi then generates a DH style key per group, and produces (n 2 1) shared keys for (n 2 1) two-party groups. In order to accomplish it, Pi generates a public key and broadcasts to the remaining group members.

The public key Xi can be generated using

Xi = g pow xi mod n
where xi is the private key of the Pi.

Each group member, Pj, where j = i also assumes a private key and generates a public key as

Xj = g pow xj mod n

where xj is the private key of Pj and 1 <= j <= n, j != i.

Each Pj then transmits Xj to the group controller, Pi. After exchanging the public keys, each member similar to the basic DH generates a unique shared key, Kj, with group controller as

Kj = (Xi) pow xj mod n = g pow xixj mod n

Similarly, Pi generates the same shared key, using

Kj = (Xj) pow xi mod n = g pow xixj mod n.

It actually generates (n-1) shared keys for (n-1) groups. The group controller combines these shared keys to produce a single group key. The Pi computes the public key Xk as given below and sends to Pj.

Xk = gPiKk!=j mod n,
where 1 _ k _ n, k = j.

Each party in the group then generates the group key K as follows
P1 generates K = (Xk) pow K1 modn = g pow(K1K2K3, ...,Kn-1Kn) modn
P2 generates K = (Xk) pow K2 modn = g pow(K1K2K3,...,Kn_1Kn) modn
P3 generates K = (Xk) pow K3 modn = g pow(K1K2K3,...,Kn_1 Kn) modn
. . . . . .
Pn_1 generates K = (Xk) pow Kn_1 modn = g pow(K1K2K3,..., Kn_1Kn) modn
Pn generates K = (Xk) pow Kn modn = g pow(K1K2K3,..., Kn_1Kn) modn
Since the group controller knows all the two-party shared keys, it also generates the group key using
K = g pow(K1K2K3,...,Kn_1Kn) modn.
The proposed Group-DH mainly consists of two steps as summarised below.

GROUP-DH TECHNIQUE:

Step 1: A member acts as a group controller and forms a two-party group with the remaining group members. Each group individually generates a DH-style key using DH technique.

Step 2: The group controller generates (n - 1) public keys by raising the exponent of g with the product of (n - 2) shared keys at a time and sends to the corresponding group members. On receiving, each member raises the exponent with its own shared key and generates the group key.

4.1.3 DES ALGORITHM:

4.1.3.1 INTRODUCTION:

This paper is the implementation of DES algorithm by which a file can be encrypted and decrypted. The encryption and decryption can be done under provision of a key. The key must be an 8 digit integer. DES encrypts and decrypts data in 64-bit blocks, using a 64-bit key (although the effective key strength is only 56 bits, as explained below). It takes a 64-bit block of plaintext as input and outputs a 64-bit block of cipher text. Since it always operates on blocks of equal size and it uses both permutations and substitutions in the algorithm, DES is both a block cipher and a product cipher. DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the cipher text. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a key using a brute-force attack. So as the number of rounds increases, the security of the algorithm increases exponentially.

4.1.3.2 SOME PRELIMINARY EXAMPLES OF DES:

DES works on bits, or binary numbers--the 0s and 1s common to digital computers. Each group of four bits makes up a hexadecimal, or base 16, number. Binary "0001" is equal to the hexadecimal number "1", binary "1000" is equal to the hexadecimal number "8", "1001" is equal to the hexadecimal number "9", "1010" is equal to the hexadecimal number "A", and "1111" is equal to the hexadecimal number "F".

DES works by encrypting groups of 64 message bits, which is the same as 16 hexadecimal numbers. To do the encryption, DES uses "keys" where are also apparently 16 hexadecimal numbers long, or apparently 64 bits long. However, every 8th key bit is ignored in the DES algorithm, so that the effective key size is 56 bits. But, in any case, 64 bits (16 hexadecimal digits) is the round number upon which DES is organized.

For example, if we take the plaintext message "8787878787878787", and encrypt it with the DES key "0E329232EA6D0D73", we end up with the cipher text "0000000000000000". If the cipher text is decrypted with the same secret DES key "0E329232EA6D0D73", the result is the original plaintext "8787878787878787". This example is neat and orderly because our plaintext was exactly 64 bits long. The same would be true if the plaintext happened to be a multiple of 64 bits. But most messages will not fall into this category. They will not be an exact multiple of 64 bits (that is, an exact multiple of 16 hexadecimal numbers).

For example, take the message "Your lips are smoother than vaseline". This plaintext message is 38 bytes (76 hexadecimal digits) long. So this message must be padded with some extra bytes at the tail end for the encryption. Once the encrypted message has been decrypted, these extra bytes are thrown away. There are, of course, different padding schemes--different ways to add extra bytes. Here we will just add 0s at the end, so that the total message is a multiple of 8 bytes (or 16 hexadecimal digits, or 64 bits).

The plaintext message "Your lips are smoother than vaseline" is, in hexadecimal, "596F7572206C6970 732061726520736D 6F6F746865722074 68616E2076617365 6C696E650D0A". (Note here that the first 72 hexadecimal digits represent the English message, while "0D" is hexadecimal for Carriage Return, and "0A" is hexadecimal for Line Feed, showing that the message file has terminated.) We then pad this message with some 0s on the end, to get a total of 80 hexadecimal digits:
"596F7572206C6970 732061726520736D
6F6F746865722074 68616E2076617365
6C696E650D0A0000".

If we then encrypt this plaintext message 64 bits (16 hexadecimal digits) at a time, using the same DES key "0E329232EA6D0D73" as before, we get the cipher text:
"C0999FDDE378D7ED727DA00BCA5A84EE7F269A4D643 8190 9DD52F78F5358499 828AC9B453E0E653".

This is the secret code that can be transmitted or stored. Decrypting the ciphertext restores the original message "Your lips are smoother than vaseline".

4.2. DESCRIPTION:

The proposed system is analysed and divided in to three modules. They are

1. Exchange of multiple two-party keys (MTK).
2. Exchange of one multi-party key (MPK).
3. Encryption and Decryption of data.

The reason for the description of modules in this manner is that we are generating the keys for two-party communication and multiparty communication. And using these keys we are performing encryption and decryption operations. That's why the system is divided in to these four modules. Let us have a brief description about each module.

4.2.1 EXCHANGE OF MULTIPLE TWO-PARTY KEYS:

In this module there will be two participants participating in the communication. One is termed as sender and the other is termed as the receiver. They are termed as so because the sender sends the data and the receiver receives the data. The data is sent in an encrypted manner. For the purpose of key generation Diffie-Hellman key exchange technique for multiple two-party key generation is used. Using this technique the parties generates 15 shared keys. These shared keys are kept secret and is not known to the third party until the secret keys that are assumed in the key generation are kept secret. These shared keys are used for encryption and decryption of data. In these 15 keys 4 keys are called base keys and the remaining 11 keys are called extended keys. The extended keys are generated by multiplying all the possible combinations of the base keys. The algorithm for the key generation is descripted in the implementation chapter.

4.2.2 EXCHANGE OF ONE MULTI-PARTY KEY:

This module is applicable for a static group. In the static group the number of participants in the group is fixed. The number of participants is fixed before the communication. It cannot be changed. In the group there will be a group controller which controls the whole group and the remaining members are called group members. The key exchange is performed between the group controller and the remaining group members. For the purpose of key exchange Diffie-Hellman technique for one multi-party key is used. Only one key is shared among the group. The communication is performed with that key only by making use of some encryption algorithms like DES. The key is kept secret until the secret values that are assumed are kept secret. The description about this module is given in the implementation chapter.

4.2.3 ENCRYPTION AND DECRYPTION OF DATA:

After performing the key exchange operation that may be the multiple two-party keys and one multi-party key the next thing is the usage of those keys. Those keys are used for encryption and decryption of data. For the purpose if encryption here DES algorithm is used. DES stands for Data Encryption Standard. DES algorithm is a symmetric block encryption algorithm. This is a block cipher algorithm. It is a private key algorithm. The plain text is 64 bits in length and the key is 56 bits in length; longer plaintext amounts are processed in 64-bit blocks. The DES structure is a minor variation of the Feistel network. There are 16 rounds of processing. From the original 56-bit key, sixteen sub keys are generated, one of which is used for each round. After performing the encryption cipher text is generated. It is not readable. After performing the encryption the sender sends the data to the receiver.

The receiver receives the encrypted data i.e. cipher text from the sender and it needs to Decrypt it. The process of decryption with DES is essentially the same as the encryption process. The rule is as follows: Use the cipher text as input to the DES algorithm, but use the sub keys in reverse order. That is, use the sixteenth sub key on first iteration, fifteenth sub key on the second iteration and so on until the first sub key is used on the sixteenth and last iteration. The key that is used for encryption only should be used for decryption. Thus the plain text is obtained.

5.RESULTS:

1. This graph shows the comparison of the Existing system with the proposed system. This graph is based upon the number of keys generated and number of users in the network.

Algorithm	Number of keys Generated	Number of users
Basic Diffie-Hellman(DH)	1	2
Multiple two-party DH	15	2
Multi-party DH	1	5

Table 5.1: Comparison of Existing System with Proposed System

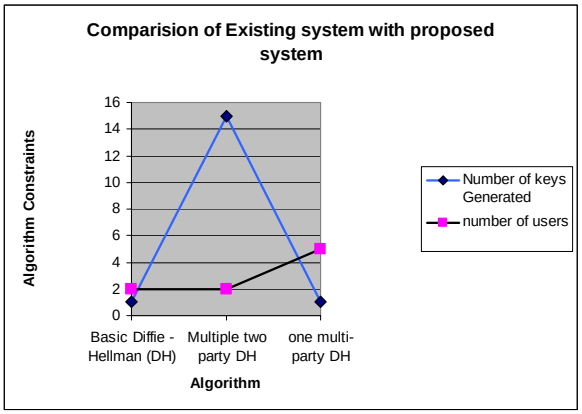

Fig 5.1: Graph showing the comparison of Existing System with Proposed System

Conf. Networks and Communications Systems, March 2006, pp. 321–

Protocols	Rounds	Messages	Unicast	Broadcast	Message size	Sequential exponentiation
CCEGK	2.32	8	5	3	8	2.64
EGK	2.32	8	0	7	8	2.64
TGDH	2.32	8	0	7	8	2.64
STR	4	8	0	7	8	8
GDH3.0	6	9	7	2	9	19
Group-DH	6	6	9	2	9	10

2. This graph shows the comparison of the proposed technique with the other group key-exchanging techniques.

Table 5.2: Comparison of the Proposed technique with the other group key-exchanging techniques

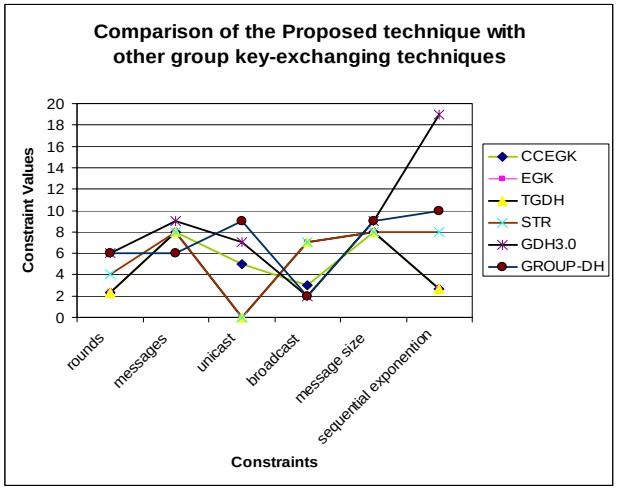

Fig 5.2: Graph showing the Comparison of the Proposed technique with other group key-exchanging techniques

6. CONCLUSION:

Two extensions of the basic DH technique are proposed and they are exchange of multiple two-party shared keys and one multi-party key. The extension proposed in the first case can generate multiple DH-style keys with comparatively less key generation overhead (KEOs). It also provides more guard to the keys and boost applicability. On the other hand, the extension projected in the second case can generate a multi-party key for large standing groups. Although

the technique is proposed for a static group, it may be easily extended for large vibrant groups. It uses simpler steps and needs comparatively less communication and computation costs, and therefore may be useful for realistic applications.

7. REFERENCE:

[1] STALLINGS W.: 'Cryptography and network security: principles and practices' (Pearson Education, 2004, 3rd edn.)

[2] STEINER M., TSUDIK G., WAIDNER M.: 'Key agreement in dynamic peer groups', IEEE Trans. Parallel Distrib. Syst., 2000, 11, (8), pp. 769–780

[3] ZHENG S., MANZ D., ALVES-FOSS J., CHEN Y.: 'Security and performance of group key agreement Protocols'. Proc. IASTED Int.

327

[4] KIM Y., PERRIG A., TSUDIK G.: 'Tree-based group key agreement', ACM Trans. Inf. Syst. Secur., 2004, 7, (1), pp. 60–96

[5] KIM Y., PERRIG A., TSUDIK G.: 'Group key agreement efficient in communication', IEEE Trans. Comput., 2004, 53, (7), pp. 905–921

[6] ZHENG S., MANZ D., ALVES-FOSS J.: 'A communicationcomputation efficient group key algorithm for large and dynamic groups', Comput. Netw., 2007, 51, (1), pp. 69–93

[7] POHLIG S., HELLMAN M.: 'An improved algorithm for computing logarithms in gf(p) and its cryptographic significance', IEEE Trans. Inf. Theory, 1978, 24, (1),pp. 106–111

[8] MCCURLEY K.S.: 'A key distribution system equivalent to factoring', J.Cryptogr., 1988, 1, (2), pp. 95–106

[9] ELGAMAL T.: 'A public-key cryptosystem and a signature scheme based on discrete logarithms', IEEE Trans. Inf.Theory, 1985, 31, (4), pp. 469–472

[10] HUGHES E.: 'An encrypted key transmission protocol'. Presented at Rump Session of CRYPTO94, August 1994

AUTHORS PROFILE

Mrs. C.Sushama has completed B.Tech in Electronics from JNT University and M.Tech in Computer Science from JNT University. Presently she is pursuing Ph.D in Computer Science and EngineeringS.V.University, TIRUPATI. She is currently working as Assistant Professor in the Department of CSE, Sree Vidyanikethan Engineering College, A. Rangampet, Tirupati, A.P. His main research interest includes Software Engineering, Software Architecture.

P.Neelima has completed B.Tech in Computer Science and Engineering from JNT University. She is currently working as Assistant Professor in the Department of CSE, Siddhartha Educational Academy Group of Institutions, C.Golapalli, Tirupati, A.P. Her main research interest includes Software Engineering, Software Architecture, Information Retrieval and Database Management Systems.

Mr. M Sunil Kumar has completed B.Tech in Computer Science & Information Technology from JNT University and M.Tech in Computer Science from JNT University. Presently he is pursuing Ph.D in Computer Science and Engineering, S.V.University, TIRUPATI. His main research interest includes Software Engineering, Software Architecture, Information Retrieval and Database Management Systems.