IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 3, May 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814


# Extended Diffie-Hellman Technique to Generate Multiple Shared Keys at a Time with Reduced KEOs and its Polynomial Time Complexity

**Nistala V.E.S. Murthy[1] and Vankamamidi S. Naresh[2]**

**[1] Department of Computer Science and Systems Engineering, Andhra University Visakhapatnam-530003,India**

**[2]Department of Computer Science,S.V.K.P. and Dr. K.S.R. Arts and Science College Penugonda-534320,India**

**Abstract**

Recently Biswas[1] extended Diffie-Hellman technique to generate multiple two-person-shared keys by exchange of two public keys. In this paper, we further generalize the Diffie-Hellman technique to generate multiple two-person-shared keys by exchange of any number of public keys and study its Polynomial Time Complexity, Security etc. Also, an upper bound for the number of shared keys in terms of the number of exchanged keys and for a given number of shared keys, the minimum required number of keys to be exchanged, were arrived at. Lastly, a comparative study between the proposed technique and the Diffie-Hellman technique repeated m-times is made.

**Keywords**: *Diffie-Hellman technique, DDH problem, multiple shared keys, key exchange operations, secure data transmission.*

## 1. Introduction

Secure Data Transmission (SDT) is one of the most important parts of Communication in these days with the advent of several transactions being e-transitioned. Interestingly, this SDT is to take place on public transmission channels like, the Internet etc. which are very much open now-a-days to every one including cyber criminals. Hence, cryptosystems were developed and are constantly under research. Some of the cryptosystem (used between two persons) involve exchange of public keys on public transmission channels and construction of a shared key in private, using private keys.

One of the logical solutions to increase security in SDT is to change the crypto-system-keys as frequently as possible. In fact, the ability to dynamically and publicly establish a session key for secured communication is a big challenge in cryptography.

Now in this direction, Diffie-Hellman developed a simple and easy to implement technique, which here onwards is referred to as D-H. At any time, between two people, it requires an exchange of one public key from each one to construct one shared/common key, using their two private keys. Biswas [1] generalized D-H to generate 15 multiple keys using two public keys each between two persons.

Now in this paper, we further generalize Biswas[1] to generate $2^{m^2} - 1$ multiple keys using $m$ public keys between two persons. The main advantages of the present work are:

**i.** The proposed technique reduces not only the computational cost significantly but also the key exchange over heads.

**ii**. Depending on the application and the security needed –by increasing the number of sessions and the number of shared keys, we can generate sufficient number of shared keys N, by selecting

$$m = \lceil |\sqrt{(\log (N+1)/ \log 2)}| \rceil$$

One might think here that multiple use of different D-H itself might solve the problem in multiple sessions. But observe that it brings in additional key exchange operations (KEOs) per session and an increase in overhead (Cf. [2]-[4]). In such conditions, if multiple shared keys are exchanged securely at a time with comparatively fewer KEOs and if a key or even multiple keys are used in the same session, it not only eases the establishment of session keys, but also reduces the key exchange overhead significantly. In what follows, first we recall the elements of Diffie-Hellman technique.

### 1.1. Diffie-Hellman technique

Diffie and Hellman [3] introduced the concept of two person key exchange technique that allows two participants to exchange two public keys through an unsecured channel and generate a shared secured key between them.

Let A and B be two participants that do not know anything about each other but wish to establish a secured shared key between them.

Then they execute the following five steps:

**Step1:** Both A and B agree on two large positive integers, n and g such that n is a prime number and g is a group generator.

**Step2:** 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.

**Step3:** Both A and B compute their public keys using $X= g^x \bmod n$ and $Y=g^y \bmod n$, respectively.

**Step4:** They exchange their public keys through a public communication channel.

**Step5:** Now both A and B compute their shared key K, using

$$K=Y^x \bmod n= g^{xy} \bmod n$$
$$K=X^y \bmod n= g^{xy} \bmod n$$

For practical applications, it is assumed that D-H holds decisional DH (DDH) assumptions (Cf. [5], [6]) which means that no polynomial time algorithm exists to compute K up on knowing X,Y, g and n.

Now, in what follows, first we outline the Multiple Shared Key technique using m public keys. We refer to this as MSK-m, Next, we arrive at (1) an upper bound for the total number of shared keys N (of course, not necessarily distinct) generatable in MSK-m in terms of the number of public keys m (2) a formula for the minimum number of public keys m required in order to generate the required number of shared keys N. Then we compare MSK-m with the repeated application of D-H m-times, establishing its polynomial time complexity. Lastly, some security aspects of MSK-m and selection/communication of shared keys is discussed.

## 2.0 Generation of $2^{m^2}$ -1 two person shared keys at a time by exchanging m public keys

In brief, each person in the proposed technique assumes m random values namely $x_1, x_2, \ldots, x_m$ and generates m public keys,$X_1, X_2,\ldots X_m$. These keys are then exchanged between the two participants and multiple shared keys are generated. The details of shared key generation are given below.

### 2.1 Multiple Shared key (MSK) Technique

The participant A generates m public keys as given below and sends to B. So, $X_i=g^{x_i} \bmod n,\ 1\le i\le m$ where $X_i$ and $x_i$, for $i=1,2,\ldots,m$, are the public and private keys of A.
Similarly the participant B chooses private keys $y_1,y_2,\ldots,y_m$ randomly and generates m public keys as

$$Y_j =g^{y_j} \bmod n,\ 1\le j\le m$$

and sends them to A.Now on exchange of m 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 exists in the basic (DH), "$m^2$" combinations such as $(x_iy_j), 1\le i\le m,\ 1\le j\le m$, exist and each of them can generate a DH style key. The generation of $m^2$ shared keys is as follows .The person A can compute the following keys:

$$K_{ij} = Y_j^{x_i} \bmod n = g^{x_i y_j} \bmod n,\quad 1 \le i \le m,\ 1 \le j \le m..$$

The person B can compute the following keys:

$$K_{ij}' = X_i^{y_j} \bmod n = g^{x_i y_j} \bmod n,\quad 1 \le i \le m,\ 1 \le j \le m.$$

Here $K_{ij} = K_{ij}',\ 1 \le i \le m,\ 1 \le j \le m$. These $m^2$ keys are called the base keys. Additional shared keys can be derived by multiplying these base keys in different combinations which are called extended keys. The extended keys are derived as follows.
For example, multiplying two base keys at a time out of $m^2$ base keys generates $C(m^2,2)$ shared keys such as

$$K=K_{11}\times K_{12} = g^{x_1y_1+x_1y_2} \bmod n$$

Multiplying three base keys at a time out of $m^2$ base keys generates $C(m^2,3)$ shared keys such as

$$K=K_{11}\times K_{12}\times K_{13}= g^{x_1y_1+x_1y_2+x_1y_3} \bmod n.$$

. . . . . .

. . . . . .

Multiplying $m^2 - 1$ base keys at a time out of $m^2$ base keys generates $C(m^2,m^2-1)$ shared keys such as

$$K = K_{11}\times K_{12}\times\ldots..\times K_{1m}\times\ldots.\times K_{m1}\times K_{m2}\times\ldots\times K_{mm-1}$$

$$= g^{x_1y_1+x_1y_2+\cdots+x_1y_m+\cdots+x_my_1+\cdots x_my_{m-1}} \bmod n$$

Finally by multiplying all $m^2$ base keys generates only one shared key

$$K = K_{11}\times K_{12}\times\ldots..\times K_{1m}\times\ldots.\times K_{m1}\times K_{m2}\times\ldots\times K_{mm},$$

$$= g^{x_1y_1+x_1y_2+\cdots+x_1y_m+\cdots+x_my_1+\cdots+x_my_m} \bmod n$$

Using the above discussion, in what follows, we arrive at the total number of shared keys at a time, in terms of the number of exchanged keys.

### 2.2 An upper bound for the total number of shared keys in the proposed technique

The total number of shared keys

= Number of base keys + The number of extended keys, which can be easily seen to be $2^{m^2}-1$

.

Observe that for m = 2, we get the 15 shared keys mentioned in Biswas[1].

**Example**

| The number of public keys exchanged($m$) | Number of base keys($m^2$) | Total number of shared keys ($2^{m^2}-1$) |
|---|---|---|
| 1 | $1^2=1$ | $2^1-1=1$ |
| 2 | $2^2=4$ | $2^4-1=15$ |
| 3 | $3^2=9$ | $2^9-1=511$ |
| 4 | $4^2=16$ | $2^{16}-1=65535$ |
| . | . | . |
| . | . | . |
| . | . | . |

### 2.3 A lower bound for m to generate the required number of shared keys

Observe that for $2^{m^2}-1 \geq N$, $2^{m^2} \geq N+1$,

$m^2 log2 \geq log\, N+1$, $m \geq \lceil |\sqrt{(log\,(N+1)/\,log2)}| \rceil$

Therefore by selecting $m = \lceil |\sqrt{(log\,(N+1)/\,log2)}| \rceil$ we can generate at least N shared keys.

**Example:** For instance if we required $N = 50000$ shared keys we find that

$m = \lceil |\sqrt{(log\,(N+1)/\,log2)}| \rceil$

$= \lceil |\sqrt{(\log\,(50001)/\,\log2)}| \rceil$

$= \lceil 0.3950907406 \rceil$

$= 4$

Thus, by exchanging 4 public keys one can generate as many as 65535 shared keys, may be not all distinct.

### 2.4. Comparison between MSK-m and D-H repeated m-times and the Polynomial Time Complexity of MSK-m

First observe that

i. To generate $2^{m^2}-1$ shared keys in D-H technique it requires $2^{m^2}-1$ rounds and proposed technique (MSK-m) requires single round

ii. To generate $2^{m^2}-1$ shared keys it requires interchange of $2(2^{m^2}-1)$ messages in D-H technique and $2m$ messages in MSK-m.

iii. To generate $2^{m^2}-1$ shared keys it requires $2^2(2^{m^2}-1)$ exponential operations in D-H technique and MSK-m requires $2m^2+2m$.

iv. To generate $2^{m^2}-1$ shared keys it requires no multiplications and MSK-m requires $2(2^{m^2-1}(m^2-2)+1)$

v. The time complexity of D-H is

$$T_{D\text{-}H}(m) = C_e 2^2 (2^{m^2}-1) = O(2^{m^2})$$

Where $C_e$ denotes time needed for execution of one exponential operation.

So, D-H possibly has non polynomial (exponential) time complexity. Hence, it requires more time for execution.

Since multiplications are very less expensive than exponentiation, for time complexity we consider exponential operations and hence the time complexity of MSK

$$T_{MSK\text{-}m}(m) = C_e(2m^2+2m.) = O(m^2)$$

where $C_e$ denotes time needed for execution of one exponential operation.

Thus, MSK-m has polynomial (quadratic) time complexity. So it requires less time for execution than D-H.

| | Number of rounds | Interchange of number of messages | Execution of number of exponentiations | Numberof multiplications | Time complexity |
|---|---|---|---|---|---|
| DH | $2^{m^2}-1$ | $2(2^{m^2}-1)$ | $2^2(2^{m^2}-1)$ | *NIL* | $O(2^{m^2})$ *(exponential)* |
| MSK | 1 | $2m$ | $2m^2+2m.$ | $2^{m^2-1}(m^2-2)+1$ | $O(m^2)$ *(quadratic)* |

### 2.5. Security of the proposed technique

We consider DDH assumption for the security of the shared keys generated using the Group-DH technique. For this we assume large finite cyclic groups that hold DDH assumption (Cf. [5]), and it is widely believed that there exist such groups for which DDH is intractable. For instance, let p, q and g be publicly known, where p and q are large primes and $p = |G| = 2q+1$, and g is the generator of a subgroup $QRp$ of $Z_p^*$ and $q = |QR_p|$. If the participants choose

their private keys from $Z_q = \{0, 1, 2. . . q -1\}$, then an adversary cannot distinguish between the random keys in *QRp* and the keys that are generated by DH technique.
In what follows, we recall some material from Zheng-Manz-Foss-Chen[7].

DDH assumption*:*
G is a finite cyclic group and g is a generator.
Given ( $g^a$, $g^b$, $g^{ab}$) and ( $g^a$, $g^b$, $g^c$) for random *a, b, c*$\in$ *[1, |G|],* no efficient algorithm can decide that c = ab in G. In other words, the value $g^{ab}$ is indistinguishable in polynomial time from a random number of G.
Using the DDH assumption, if $K_1$ and $K_2$ are random numbers and $PK_i$ for $i = 1, 2$ are the corresponding public values, then the shared value $K = PK_1 \; exp \; K_2 = PK_2 \; exp \; K_1$using two-party secure group key exchange is indistinguishable in polynomial time from a random value, where exp is an exponentiation operation. From the security point of view, the above assumption is very strong and many secured practical cryptographic systems designed are based on it. The present paper also follows this assumption to generate secured multiple two-party shared keys.
Now we make the following Proposition.

**2.5.1 Proposition:**
$m^2$ two-party shared keys $k_1, k_2, k_3 \ldots\ldots k_m{}^2$(base keys) derived in the application of the basic DH technique are indistinguishable in polynomial time from random numbers.

**Proof:** Since each of the $m^2$ shared keys is basically a D-H style key and for D-H shared key the proposition is true, our Proposition follows.
**Corollary 1:**
The extended $2^{m^2} - 1 - m^2$ shared keys generated by multiplying the $m^2$ base keys in different combinations are also indistinguishable in polynomial time from random numbers.
For a communication, the participants must agree upon a particular key. One method for the selection of a shared key is shown below.

**2.6 Selection of shared key**

Now that the proposed extension can generate multiple shared keys, it is necessary for us to be able to select a key for a session. Here we suggest that one can follow a method similar to that in the well known Merkle's puzzle which is recalled below:
A party generates n messages each with having a different puzzle number and a secret key number and sends all the messages to the other party in encrypted form. Note that a different 20 bit key is used for encryption of each message. The other party chooses one message at random and performs brute-force attack to decrypt it -although it needs a large amount of work, it is still computable. It then encrypts its message with the key thus recovered and sends it to the first party along with the puzzle number. Since it knows the puzzle number, it thus identifies the key and decrypts the message.
Similarly, in order to select a key out of $2^{m^2} - 1$shared keys, one party generates a message comprising a shared key and a puzzle number. After encrypting it either with the smallest or largest key of the shared keys generated, it is sent to the other party. The message is easily decrypted as the recipient knows all keys and the (shared key, puzzle number) pair is recovered. The party then either sends the puzzle number alone or an encrypted message along with the puzzle number to the first party, where the message is encrypted with the shared key found. Since the first party knows the puzzle number, it therefore identifies the session key and can decrypt the message. For subsequent changes, the present session key may be used to encrypt a shared key at one end, and it is decrypted at the other end to obtain the new session key.
**Conclusions:** Since the D-H technique is simple and easy to implement and since MSK-m uses only the D-H style for shared key generation, MSK-m is also easy to implement. Further, using the lower and upper bounds for m and N (Cf. 1.2 and 1.3), the security levels can be increased in SDT with relatively lesser operational overhead (Cf.1.4).

**Acknowledgments**

Author would like to thank the Management of **S.V.K.P. and Dr. K.S.R. Arts and Science College**, for Sponsor ing and financial support.

**References**
[1] G.P. Biswas, Diffie Hellman Technique Extended To Multiple Two Party Keys And One Multi Party Key, IET inf. Sec., 2008, Vol.2(1), pp.12-18.
[2] Menezes A.J., Elliptic Curve Public Key Crypto Systems, Kluwer Academic Publishers, 1993.
[3] Diffie W. and Hellman M., New Directions in Cryptography, IEEE Trans. Inf. Theory, Vol. 22(6), pp. 644–654, 1976.
[4] Stallings W., Cryptography And Network Security, Principles And Practices, Pearson Education, 3rd Edn., 2004.
[5] Boneh D., The Decision Diffie-Hellman Problem, Proc. 3rd Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, 1423, pp. 48–63, 1998.
[6] Boneh D. and Venkatesan R., Breaking RSA May Not Be Equivalent to Factoring, Advances in Cryptology, EUROCRYPT'98, pp. 59–71, 1998.
[7] ZHENG S., MANZ D., Alves-Foss J and Chen Y., Security And Performance Of Group Key Agreement

Protocols, Proc. IASTED Int. Conf. Networks and Communications Systems, pp. 321–327, March 2006.
[8] Merkle R.C., Secrecy, Authentication And Public Key Systems, Communications ACM, Vol. 21(4), pp. 294–299, 1978.

**About the Authors**

**Nistala V.E.S. Murthy** is currently working as a Professor in the department of Computer Science and Systems Engineering of Andhra University, Visakhapatnam. He developed f-Set Theory –wherein f-maps exists between fuzzy sets with truth values in *different* complete lattices, generalizing L-fuzzy set Theory of Goguen which generalized the [0,1]-fuzzy set theory of Zadeh, the Father of Fuzzy Set Theories. He also published papers on Representation of various Fuzzy Mathematical (Sub) structures in terms of their appropriate crisp cousins.

**Vankamamidi Srinivasa Naresh** is currently working as a Director, for the Post Graduate Department of Computer Science Courses in **S.V.K.P. and Dr. K.S.R. Arts and Science College**. He obtained an M.Sc. in Mathematics from Andhra University, an M.Phil. in Mathematics from Madurai Kamaraj University and an M.Tech in Computer Science and Engineering from J.N.T. University-Hyderabad. He is also a recipient of U.G.C.-C.S.I.R. JUNIOR RESEARCH FELLOSHIP and cleared NET for LECTURERSHIP.