VIT

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Project Report for J component on

Topic:

Approach to Improve Diffie–Hellman Key Agreement algorithm for providing high security to clients with limited Computational Resources

Network and Information Security - Embedded Project

ITE4001

Under the Guidance of

Prof. ASWANI KUMAR CHERUKURI

SLOT:F2+TF2

Fall semester2020-2021

By

Aman Chawla 18BIT0055

Pranav Bansal 18BIT0244

Lakhan Nad 18BIT0182

Shaurya Gupta 18BIT0068

Problem Definition:

The modular exponentiation computation of Diffie-Hellman key Exchange Protocol puts a heavy load on the clients with limited Computation resources like Radio Frequency Identification (RFID), NFC, Arduino, Other Embedded Systems, where either the **computational capacity** or the **battery is limited** and precious and cannot be intensively used for Security Computations which would hinder the required performance of the system.

There-fore, many standards and implementations of these thin clients opt out key agreement schemes to meet computational efficiency requirements thereby accepting some security risks, but with increase in the usage of these embedded systems, It is important to have them more secure and efficient algorithm that does not hinder the actual performance and application of the device.

Precise problem statement reflecting your project:

Our proposed project aims to **Implement, Test Validity, Feasibility and Practical Application of a Proposed Key Sharing Algorithm based on Diffie-Hellman Key Exchange Algorithm for Thin clients with resource constraints.**

The Proposed Algorithm in the Research paper is Computationally Modified form the Existing Algorithm
Some Important facts:

- The modified Computational Diffie–Hellman Problem and prove its security being equivalent to the Computational Diffie–Hellman Problem along with keeping the fact in mind that the algorithm needs to be having very limited computational overhead as it has to be implemented in devices with limited computational capabilities ,but they require high security.
- Our project work and research work is inspired by a research paper titled, A Generic Approach to Improving Diffie–Hellman Key Agreement Efficiency for Thin Clients, form The Computer Journal Advance Access published November 2, 2015.

<u>List of Objectives:</u>

1. To **find a solution** for implementing a improved Diffie-Hellman Key Agreement algorithm to improve the security of Embedded Systems and IoT devices with limited computational resources by reducing computational cost at client side.

2. To **Implement** the computationally modified Diffie-Helman Algorithm measure both Qualitative and Quantitative properties, and look for any unexpected anomalies that may occur and hinder the performance or the application of the system.

3. **Testing the results** which include:

   Testing the Efficiency of the implemented algorithm, test it for different environments and compare the modified DH algorithm with existing algorithm for its application in thin clients with limited resources in terms of time required to execute and memory required for execution and establishment of a shared key.

Techniques to be Used & Experimental Setup:
Identify the techniques & experimental setup (S/W & H/W) you wish to use to address the list of objectives.

Techniques:

- Diffie-Hellman algorithm for key exchange
- Number Theory involving exponents and Modular arithmetic
- Bitwise operations
- Linux socket programming which can be used for secure key exchange between two computers provided we know the IP addresses of both the computers

Experimental Setup:

- The configuration of the system would be Intel Core i3-4005U, 64-bit Windows 10 OS,1.7GHz, 8 GB RAM for testing the results.
- Virtual environment client for linux socket programming
- EDH-SC protocol in C language
- The devices should have computational capability to perform DHKEP and have an input/output interface

Research gap:

We found the research gaps that affects the way we proceed for implementation of the project objectives:-

1) The implementation and analysis results of key exchange approach using modular arithmetic in multiparty key exchange algorithm, is not found.
2) The implementation and analysis results of key exchange approach using modular arithmetic in embedded systems is not discussed.
3) The Implementation of modular arithmetic based key exchange algorithm with string based authentication is not found, which results in need for a statistical analysis to be carried out for observing the ratio of time reduction/optimization in Diffie-Hellman key exchange algorithm
4) The application of string based authentication mechanisms in the multiparty key exchange algorithm, because, authentication mechanism has a prominent effect over the time complexity of mechanism, and that when over an iterative approach algorithm, increases the time complexity to a great extent.

In the given literature, these results of execution of string based authentication mechanism and their analysis are not covered, which further makes it difficult to observe the influence of this authentication mechanism over a 2-party key exchange algorithm.

5) In Arazi's approach of replacing the message in the Digital Signature algorithm with Diffie-Hellamn key exchange algorithm, there are certain vulnerabilities which include a dynamic problem of, if one of the session keys are exposed in public, all the other secret session keys are compromised with it, there were many solution towards this problem, but none of them till our research scope were able to produce secret session keys which are modular in nature and independent of each other. We take this as opportunity to take our research to help make independent keys, this could be further done making the changes in the mathematical approach that standard Diffie-Hellman key exchange protocol follows.

Literature Survey:

Reference Number	Title of the Paper	Technology Used	Brief Description about the model/system	Parameters influencing the performance of the model/system	Advantages of the model/system	Limitations of the model/system
[1]	Research on Diffie-Hellman Key Exchange Protocol	Using hashing algorithms to provide entity authentication to prevent man in the middle attacks during key exchange in Diffie-Hellman Key Exchange protocol.	Comparing the efficiency of Hashing algorithms(MD5,SHA-256), symmetric key encryption algorithms(DES, AES) and public key encryption algorithms(RSA, ECC) on the basis of computations and time taken. Applying the most efficient algorithm(Hashing algorithm) on Diffie Hellman Key Exchange protocol to provide entity authentication so that participating parties are authenticated and their communication channel is secure.	Performance of the system is influenced by the choice of algorithms: Hashing algorithms are the most efficicient ones. Then comes symmetric key encryption algorithm at the least performance in terms of time taken is by the use of public key encryption algorithm. All these authentication algorithms are used over Diffie hellman key echange protocol. So the efficiency does not depends upon the protocol	The proposed method can prevent man in the middle attacks. The proposed method can prevent impersonation attacks	The model is still vulnerable to clogging attack. The advancement in computation power of devices has made brute force attack easier on various existing hashing algorithms. As a result, some hashing algorithms are vulnerable to brute force attacks.
[2]	A study on Diffie-Hellman Key Exchange Protocol	Authenticated Key establishment Protocols along with One-Pass Protocols are used to improve security of Diffie Hellman Key	Introduction : The purpose of Diffie-Hellman protocol is to enable two parties to securely exchange a session key which can then be used for next symmetric encryption of messages. The idea of Diffie-Hellman protocol is to calculate a session key by the	Performance of the system is influenced by the choice of authenticated key protocols . Considering the key exchange or	1. Diffie-Hellman's effectiveness comes from the difficulty of calculating discrete logarithms 2. Using authenticated key mechanism with asymmetric techniques	1. The protocol can only be used for exchanging secret data without authenticating two parties. This is the reason why Diffie-Hellman is insecure

		Exchange Protocol during key exchange mechanism . In addition to that , to prevent from MITM attacks , vulnerability can be avoided with the use of authentication mechanisms such as digital signatures and public- key certificates.	communicating entities based on public parameters that are shared in the initial phase. This type of protocol is called key agreement protocol. Details: Station-to-station (STS) protocol where two-party authenticated key exchange occurs , the legitimate parties can compute a secret key using Diffie-Hellman and then authenticating each other by exchanging their digital signatures.This has improved the secrecy by using short-term keys for subsequent sessions	establishment protocols from two perspectives: cost/efficiency and security are primary parameters influencing the performance . Cost includes both processing and communication costs. Also , Redundancy in communication affects the performance of system .	, we can reduce the scope of Key compromise impersonation (KCI) attack and Ephemeral key compromise (EKC) attack. 3. Biggest advantage of One-pass key exchange protocols is that they are authenticated implicitly , hence ,no one else can learn this value except the communicating parties	against man-in-the-middle attack. 2. Drawbacks of one-pass protocols include not supporting perfect forward secrecy (PFS), lack of key control and pruning to key compromise impersonation (KCI)attacks.
[3]	An Efficient Improved Group Key Agreement Protocol Based on Diffie-Hellman Key Exchange	Introduction of a managing node to gather and dispatch information of generating group key, therefore reducing the time complexity for dealing with paroxysmal needs for group key among multi nodes to O(N).	Selecting one node from multiple nodes as managing node (usually with the best computational capability) which will maintain the complete topology structure of a complete binary tree . It will reduce redundant operations while dealing with non managing nodes.The time for managing binary tree will be significantly reduced. The number of times of sending data packages to minimum. Managing node selects necessary part of information from complete binary tree to each non-managing node.	Selection of managing node will effect the performance of the system. Node with the best computational capability should be selected, and if they are of same capabilities, then the one with the lower IP address should be selected.	It will reduce the the time to manage the binary tree significantly, thus becoming a better option in case of dealing with paroxysmal needs for group key among multi nodes. It can effectively reduce overlapping computing, decrease the packages sending times. It can prevent traditional man in the middle attack	Theoritically, this method consumes more time when dealing with dynamic addition and deletion. The protocol is vulnerable to improved man in the middle attack in which the MITM hijack all the communicating path between managing and non-managing nodes

[4]	Secure E-passport Protocol using Elliptic Curve Diffie-Hellman Key Agreement Protocol	The method is based on the On-Line Secure E-passport -(OSEP)protocol where use of the Elliptic curve Diffie hellman Key Agreement protocol is done to define the session key. Further , fingerprint template is used to compute an elliptic curve to secure the combination between the chip and the IS. A mutual authentication between the RFID chip and the Inspection System takesplace.	1.There are mainly 3 steps. 1. Initialization phase : We need to define an elliptic curve using minutiae's coordinates. 2. IS Authentication : We define the session key that will be used to secure the communication between the E-passport chip and the Inspection System(IS) in the visiting country. 3. E-passport holder Authentication : Here IS will check that E-passport holder is genuine and not a malicious traveler. In addition to above ,we create an ideal elliptic curve for cryptographic use. This elliptic curve will help chip and IS to define session key. An authentication protocol is run between the chip C, IS and Document Verifier (DV). These entities have public and private keys certified by DVs. If successful, we proceed to authenticate the holder.	1. To perform verification for the E-passport holder , parameters influencing the performance are Public parameters which are defined from the Biometric Templates (BT). For example : Fingerprint 2. Also , the Document Verifier DV chooses a prime number p. Then, it generates an elliptic curve E. He chooses also a point P $\in$ E which will be the public point for the chip. Thus , following are influencing parameters: : 1. ID: identifier; 2. p: the prime number; 3. P: the first public point; 4. A and B the coefficients of E DV will also add the conventional parameters for the E-passport like name, country, age, gender…	1.The Elliptic curve key length is shorter than RSA (Rivest Shamir Adleman) key length. We conclude that ECC outperforms RSA Algorithm 2. Use of existing ICAO PKI implementations (first generation Epassports),the performance of this solution is better than OSEP because it uses elliptic curve cryptography, the keys are shorter 3. Another advantage is the use of RFID tag which has less memory. 4. The border security check-point or the Document Verifier DV can check if the traveler is in a black list of criminals.	1.Weaknesses in the OSEP protocol :The first threat, the authors wanted to replace the use of the key, used in Basic Access Control BAC, because keys have insufficient entropy. 2. The second threat is due to OSEP being an automated protocol where there is no human role; one malicious traveler can use a stolen E-passport and presents it to the IS , because OSEP protocol only checks if the E-passport is genuine and not his holder. 3.The fuzziness of BT can lead to false rejection of genuine travelers

[5]	Use of Digital Signature with Diffie Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing	In cloud computing various protocols are used to ensure security and privacy of data downloaded or uploaded to the cloud. This paper uses Diffie Hellman key exchange along with AES encryption and digital signatures and proposes one such way of data security in cloud based services and platforms.	Using this method we can incorporate three data security control mechanisms like security encryption and verification in a stand alone system. It is a three ways protection scheme wherein digital signature provides authentication, encryption algorithm provides session encryption key and is used to encrypt user data file as well, which is to be saved in cloud and lastly trusted computing to verify integrity of user data. When a user wants to upload a file to the cloud server, first key are exchanged using Diffie Hellman key exchange at the time of login, then the client is authenticated using digital signature. Finally user's data file is encrypted using AES and only then it is uploaded to another (cloud) Storage server.	There are various parameters that can influence the performance of the system. The size of file is one such factor. If size is too large then encryption/decryption of data as well as upload or download would take a lot of time and since our data is first made available to trusted server then to us this time can play a major role in deciding system's performance	The success of this system lies in fact that it avoids modification of user's data in cloud environment and prevents data to be read by encrypting the data and hence the user's data is safe and cannot be tampered. Another benefit of using this method is safe communication between server and user because of Diffie Hellman key exchange all the communication is done over a secure channel.	The limitation of this system is requirement of two servers for processing this can increase hardware cost. Since the data is first uploaded/downloaded to trusted server for encryption it takes twice as much as time to complete job than it would have been taken by one server. If security of trusted server is compromised then the whole data is exposed and risk of data tampering or data theft becomes unavoidable.

[6]	Efficient Implementation of Diffie-Hellman (DH) Key Distribution Algorithm in Pool-Based Cryptographic Systems	In this paper three different methods are proposed to efficiently use the True Random Numbers Pool while using the pool for Diffie Hellman generation algorithm. The three algorithm had same goal generation of x and y random strings based on given n such that x,y < n as used in Diffie Hellman key exchange.	The first algorithm generated the keys of length k where k is log2(n) and if the resultant random number is less than n then the algorithm is terminated or else continued. The second algorithm generated first x significant bits such that they are less than the first x bits of n and till such bits were not found the algorithm continued. Remaining bits needed to be generated only once and it was ensured that overall random number is less than n. In third algorithm one limitation of the second algorithm was addressed. Due to implementation of second algorithm it was ensured that the resulting no is never equal to n. But third algorithm first generated x significant bits keeping all other bits zero and ensuring that resultant no is less than n and remaining bits were generated randomly and if the result was greater than n then process repeated.	The only factor that can affect the performance of model or system is the speed of generation of random bits by True random number generators and the speed by which the pool of bits is consumed. Though the proposed method efficiently uses the generated random numbers with minimal wastage of random bits. But it is not ensured that bits are not wasted.	If this method is used while implementing the DH key exchange algorithm it can ensure the speed in generation of keys. The proposed algorithm cut short the waste of random numbers and hence achieving the efficient use of pool of TRNs. The proposed algorithm follows all property of random numbers with less wastage.	The only limitation of this algorithm is that this algorithm can minimize the wastage but doesn’t actually ensure that bits are not wasted. On test performed using this system it was found that this system also had wastage of around 1.69% of random bits but was significant improvement over more traditional way in which the waste was around 20%.

[7]	A Parallel Key Generation Algorithm for Efficient Diffie-Hellman Key Agreement	The paper proposes a novel key generation algorithm which achieves computational efficiency from constructing a parallel architecture.	The main idea of the parallel algorithm is to separate modular exponentiation over finite field p into s portions where each portion is bounded with k. The algorithm consists of three stages. In first stage we create a residue table in advance. Then we compute s portions simultaneously. At last we multiply all s portion to get the final key part. The main strength of algorithm lies in fact that s portions are independent of each other and can be computed parallely.	The parameters influencing the calculation is k i.e. chunks in which the original exponent is divided by choosing suitable value of k the speed of algorithm and hence overall efficiency of proposed method can be increased.	One benefit of proposed system is fact that it reduces the computational power which is required for serial key computation. One another benefit that can be obtained from this system is that if the time required to generate keys is reduces it improves the overall time required by Diffie Hellman algorithm and key exchange is fast.	The algorithm proposed only increases efficiency if performed in parallel which require more process threads to be given to the process. This increases hardware load.
[8]	Group Diffie Hellman Key Exchange Algorithm Based Secure Group Communication	The algorithm is proposed is for group communications. This algorithm proposes a way to generate a common key for group meeting which changes with change in membership of meeting. That means a new key is generated when meeting starts and whenever someone leaves or when new or old member joins the meeting. This ensures that	The algorithm is divided in two parts. One is up flow and other is down flow. As in any other DH key exchange two numbers are chosen one prime prime no and other primitive root of chosen prime. A set is maintained X={Y,Z} where Y itself is a set of numbers. The up flow starts from User1 and flows to other users until it reaches to end user. When a set reaches a user he exponents each element in Y with its private key and then concatenate Z with Y to form new Y'. He also exponents Z with it's private key to form Z'. This form his new set X' = {Y', Z'}. Down flow starts as this set reaches to last user. In down flow the last user broadcasts the Y counterpart to all other users. After that each user	The parameters influencing the calculation is number of people involved. Since the DH calculation is already time consuming Other factors are dynamics of membership. As new members join or existing members leave the group the key needs to be recomputed and hence performance is highly dependent.	A major benefit of this system is it enables secure communication among group. Because new keys are generated whenever group constituents changes it maintains forward and backward secrecy.	One major drawback of this system is that key generation is time consuming since it hops from one user to other it almost takes n steps for secret key generation where n is no of users. Every time group constituent changes the key needs to be recomputed for all the users which again make this process computationally inefficient.

		meeting channel is secure and safe and ensures both forward and backward secrecy.	extracts a element that doesn't contains only his private key counterpart and makes the secret key.			
[9]	Hybrid Technique of Genetic Algorithm and Extended Diffie-Hellman Algorithm used for Intrusion Detection in Cloud	In this proposed method, the hybridization of the Genetic Algorithm and extension of the Diffie Hellman Algorithm is used to come across the optimized path for data transmission in insecure network. Without getting interrupted by any intruders it may gives an optimized path without data damaged or without getting server busy transmission may continuous and secure. In following steps:	We will discuss, in this paper about the hybridization of Genetic algorithm and extension of Diffie Hellman algorithm try to get an idea to obtain a secure path for data transmission in insecure network without having prior knowledge of each other. It may help to identify the intruders which are inevitable and may affect the whole transmission or ma affect the data in between the transmission.	1.Selection: In this process, a population of individuals is created. And from that selection of best group is been made for the further. 2. Crossover: In crossover process, from number of groups only two chromosomes are selected for crossover to generate a new individual. 3. Mutation: After the crossover process, mutation is performed to the new individual by changing any value of it so that it seems to be different from other individuals. 4. Fitness Function: Fitness function is check to see whether the obtained individual is up to our desired goal or not	1)Genetic algorithm and extension of Diffie Hellman algorithm give a most optimized path for data transmission in insecure network by detecting intruders before it disrupts the transmission or data. 2) By using this algorithm it do not allow any intruders to make the server busy. 3) In future scope it can have an enhanced implementation of our current proposed model with other algorithms.	1. Lack of authentication procedure. 2. Algorithm can be used only for symmetric key exchange.
[10]	Modification of Diffie–Hellman Key Exchange Algorithm for Zero Knowledge Proof	In this paper zero-knowledge proof, Fiat-Shamir ZKP Protocol and D–H key exchange algorithm has been presented and criticized.	There are networks and entity groupings that require entity authentication while preserving the privacy of the entity being authenticated. Zero-knowledge proof (ZKP) plays an important role in authentication without revealing secret information. Diffie–Hellman	The Diffie-Hellman algorithm is susceptible to two attacks; the discrete logarithm attack and the man-in-the-middle attack . A Discrete Logarithm attack An interceptor	1)The proposed protocol is a deterministic algorithm with bounded values (not probabilistic), hence has no soundness error and no additional	1. Encryption of information cannot be performed with the help of this algorithm. 2. As it is computationally intensive, it is

		Two versions of the proposed protocol are presented; the first one was built around the basic D-H key exchange algorithm, which is vulnerable to man-in-themiddle-attack. The second proposed version solves the problem of the mentioned attack.	(D-H) key exchange algorithm was developed to exchange secret keys through unprotected channels.	(Eve) can intercept R1 and R2 and Find x from (R1 = g x mod p); Find y from (R2 = g y mod p); Then she can calculate (K = gxy mod p). The secret key is not secret anymore. To make Diffie-Hellman safe from the discrete logarithm attack, the following are recommended: 1) The prime number p must be very large (more than 300 digits). 2) The generator g must be chosen from the group . 3) The numbers x and y must be large random numbers of at least 100 digits long, and used only once (destroyed after being used). Still, no algorithm for the discrete logarithm problem exists with computational complexity O(x r) for any r; all are infeasible.	computation cost. C. 2) The proposed protocol fulfills the ZKP properties and protected against discrete logarithm attack and man-in-the middle attack. 3) The proposed algorithm serves as key exchange algorithm with the addition to authentication services.	expensive in terms of resources and CPU performance time. 3.
[11]	Security Analysis and Improvement for Kerberos Based on Dynamic Password and Diffile-Hellman Algorithm	Kerberos authentication system is a current protocol widely applied in the authentication mechanism in the distributed computing environment, and	This paper analyses the process of Kerberos authentication protocol, and finds the existing security problems of the protocol referring to some Kerberos improvement projects proposed in some papers at home and abroad, and puts forward the thoughts and methods using dynamic password to improve the encryption security during the	Diffile-Hellman algorithm is one of the key security exchange mechanism, it can ensure that keys need to exchange would be safely passed through unsafe network environment. The security of its origin is	1) It mainly proposes some improved methods to solve the problems of the dictionary attacks, the efficiency of encryption and decryption and the cost of key storage.	1) The replay attack: In order to avoid replay attacks, the protocol uses the time stamp mechanism in the tickets and the authentication operator. 2) Dictionary attack:

		it makes the unsafe factor, which exists in the authentication process, transfer from the distributed workstations to the centralized authentication server, providing identity authentication for the two main bodies in the open network environment and encrypting each session using session key distributed by the authentication center	process of interaction between the client and Kerberos key distribution center, and makes passwords securely exchanged by using Diffile-Hellman key algorithm.	that its mathematical model is known as a big problem in mathematic field. The object of mathematics model is calculation of discrete logarithm in finite domain, and this calculation is more difficult than exponent calculation. Its mathematical calculation process could be described as follows: (1)A chooses random number x, and calculates ax mod p, and sends the results to B; (2)B chooses random number y, and calculates ay mod p, and sends the results to A; (3)A calculates K1 = (ay) x mod p, B calculates K2 = (ax) y mod p. Through the process of these three steps, K1 and K2 both equal the result of caculation axy mod p. Because the eavesdropper can only sniff the value of a, p, ax mod p and ay mod p, they can't calculate the value of K1 and the value of K2, unless they can calculate discrete logarithm to resume the x and y	2)It put forward aim to guarantee the security of users key under the premise of maintaining effective, and to reduce the maintenance cost of keys storage	In step (2), the message sent from AS to C is encrypted by users private key which is produced with user password, and the users private key is dealt with user password by using one-way Hash function. 3) Key storage problem: Traditional Kerberos uses symmetric cryptosystem, so the shared keys between customers and KDC, between application server and KDC, between KDC and distant KDC have to be established and maintained

[12]	Group Key Exchange Protocol Based on Diffie-Hellman Technique in Ad-Hoc Network	Analysis of Tseng's protocol (Y.-M. TSENG, T.-Y. WU, INFORMATICA 2010, 21, 2, 247-259) is done. The Tseng's modified protocol is implemented and tested for ad-hoc WLAN with 3, 4, and 5 nodes.	One of the most important issue for secure communication in wireless networks is generating a common key for many participants. We show that Tseng's protocol fails to establish a common key when the key generated by Diffie-Hellman protocol is not invertible. It is modified so that its correct work is guaranteed, security features are preserved, and its computational complexity in the most time consuming controller part is decreased at least three times.	In order to provide security, working with big numbers is needed thus the parameter q that is considered in this study is a 100-digit prime number and the primitive root @(alpha) is a 50 digit number. Moreover, a secret key Xa used in the exponent of the Diffie Hellman method is a 50- digit number.	1)Establishing a common shared key for all members of the ad-hoc network or Wireless Sensor Networks (WSN) is safety critical and needs low computationally complicated protocols because of power and memory restrictions 2)Tseng's protocol is modified so it is guaranteed it's correct output, and the output producing takes at least three times less time for the controller part versus the original one. 3)The Tseng's modified protocol preserves security features of the original one.	1.As there is no authentication involved, it is vulnerable to man-in-the-middle attack 2.Digital signature cannot be signed using Diffie-Hellman algorithm
[13]	Secure Key Exchange Using Enhanced Diffie-Hellman Protocol Based on String Comparison	A variant of Diffie -Hellman Key exchange protocol based on string comparisons that addresses the trivial vulnerabilities in Diffie Hellman Protocol of authenticating the users in the communication	The proposed key exchange protocol enables two users to securely agree on a secret key with a less computational cost and little overhead for mutual authentication. The protocol is divided into three phases: initialization ,exchange, andverification. In the verification phase, both the users A and B generate verification strings. If the strings SA and SB match, then	With the increase in the number of bits(k) and size of prime p, the execution time increases andit increases drastically at. As the security of protocol improves on large value of |p| and k, so there is a trade off between security and cost of computation. For lengthof|Na|≥80,	Addresses the security vulnerabilities identified in the trivial Diffie-Hellman Protocol. It uses a combination of commitment scheme and authentication strings towithstand man-in-the-middle attack and can be used in both-wired as well as wireless networks probability that an	Probability that an attacker can launch an attack is2^(k+l)at maximum, where k and l are length of the USER_ID and the random string respectively, used for mutual authentication

			both the users A and B accept each other's DH-parameters	authentication message contains mostly 0 sand 1 s.	attacker can launch an attack is2^(k+l)at maximum, where k and l are length of the USER_ID and the random string respectively, used for mutual authentication	
[14]	Experimental Study of Diffie-Hellman Key Exchange Algorithm on Embedded Devices	This paper gives an insight into deployment of diffie-Hellman Algorithms in Embedded Systems like Audrino,RPI etc.	Diffie-Hellman based cryptography implementation on two embedded devices communicating with each other over Ethernet. During the attack the attacker uses buffer overflow problem to overwrite the program address stored and run the malicious code.	Diffie Hellman works efficiently for Arduino and Raspberry when parameters selected are within in a definite range if direct method of calculation is applied. To enable diffie Hellman algorithm for large values and secure key generation, we need to simplify the algorithm calculation. we simplify it to make it memory efficient and faster. Therefore we go ahead with an approach called fast modular exponentiation.	This paper is based on implementation of Diffie-hellman key exchange algorithm that helps to share the private key between 2 systems. This Implementation lets iot and embedded systems communicate the keys securely for further communication. This introduces more abstraction in an assembled system and helps implement better security while exchanging data.	The work involves a pre assigned port number (telnet port 23) , over which a server listens to ; hence there is a possibility of data overriding by a third party which will be using that port number within the same network. Usually the value of q and p is taken large to ensure better security and direct method of Diffie-Hellman algorithm calculation would result in values out of the memory limits of Arduino and Raspberry pi
[15]	Efficient Extended Diffie-Hellman Key Exchange Protocol	The proposed key exchange algorithm uses modular multiplication instead of exponential	The proposed group key agreement protocol is designed to provide the key agreement among the group members the divide and conquer strategy is used for reducing the computation cost. If there are n members in the group then total	We have considered three parameters like exponentiation, multiplication, and number of steps for comparing the proposed algorithm and the Diffie	It can be easily derived that the increase in time in the case of the Diffie-Hellman is more as compared to the increase in time in case of the proposed	The algorithm is proposed with slight decrease in security and but can be very helpful for systems with low configuration.

		powers which make the key distribution process slower.	O(log2n) iteration requires. The group of members shares the common session key among them for which the proposed technique applied the divide and conquer technique for sharing the public keys and establishing the common session key among them in less time in which we apply the following processes are swapping blinded keys, modular multiplication, and obtaining the secret we add the private key on the resultant. Each group member chooses a random private key, and the public keys are exchanged by the group for generating common session key.	–Hellman Key Exchange protocol.	approach. the proposed algorithm is fasterthan the classical Diffie-Hellman algorithm and requires less computational time than the Diffie-Hellman algorithm. The proposed algorithm takes(2*n−2)number of steps where Diffie-Hellman required n^2	
[16]	Integrating Diffie–Hellman Key Exchange into theDigital Signature Algorithm (DSA	Three different protocols that securely integrate DH key exchange into DSA for authenticated key distribution	This protocol supports non-interactive applications, such as secure e-mail transmission. Let us say user A wants to securely send an email to user B. Key K(a ,b) is the shared secret key This protocol supports interactive applications. Let us say user A wants to communicate with user B interactively. Here, Kab and Kba are the shared secret keys for directions A to B and B to A, respectively. Three-Round Protocol supports interactive applications. shows the three-round protocol. Here, user A wants to communicate with user B interactively. This protocol differs from the two-round protocol in that it includes key confirmation in the third round. Here k(a, b) and k(b, a) are the	User B confirms with user A of receiving the shared secret key K(a,b). by signingthis key along with m(b), Since this shared secret key K(a,b), a function of a random integer that was originally selected by user A, acts as a unique checkpoint, after receiving the signature from user B, user A is convinced that the message is not a replayed one and knows that it is indeed from user B, Thus it clarifies that K(b,a) is not a replayed key.	We show that the three-round protocol can prevent known key attack , key replay attack and unknown key-share attack. given one of the shared secret keys (e.g.,K(a,b)) , finding the other shared secret key,K(b,a) is no easier than a discrete logarithm problem by including the shared secret keysin the signature equation we prevent known key attack in our protocol. The shared secret key	Limitations are subject to Qualitative reports when the algorithm is tested heavily in fabricated environment

			shared secret keys for directions A to B and B to A respectively.		is included in the signature equation along with the message in our scheme. This arrangement prevents the known key attack and the key replay attack. three-round protocol achieves key confirmation, which prevents the unknown key-share attack.	

Reference Number	Title of the Paper	Technology Used	Brief Description about the model/system	Parameters influencing the performance of the model/system	Advantages of the model/system	Limitations of the model/system
[17]	A Generic Approach to Improving Diffie–Hellman Key Agreement Efficiency for Thin Clients	Algorithm comparable to Authenticated Diffie–Hellman key (D-H key) agreement for establishing secure session keys for Security in Embedded system.	The paper proposes general technique to enhance the computational load for thin clients when establishing secure D-H keys. The proposed approach achieves the same security of conventional authenticated D-H key agreement except the perfect forward secrecy.	The major Parameter which on which the whole topic isdependent is the limitation of computation resources either due to hardware limitation or Power limitation. Due to which it has to be kept in mind to provide high security like Algorithms with large computation overhead along with lower Time and Space Complexity and lesser Computation Overhead.	The given approach reduces one modular exponentiation for the client, but adds one more exponentiation on the server. This approach improves client efficiency at most 50%when it applies on the original D-H key agreement scheme. The improvement might seem insignificant or just a trade-off for conventional computer-to-computer interaction, but it does matter when it is applied on those thin clients where either the computing resources or the batteries are limited	As the proposed generic approach does not own perfect for-ward/backward secrecy, some mechanisms could be designed to mitigate the possible risks.

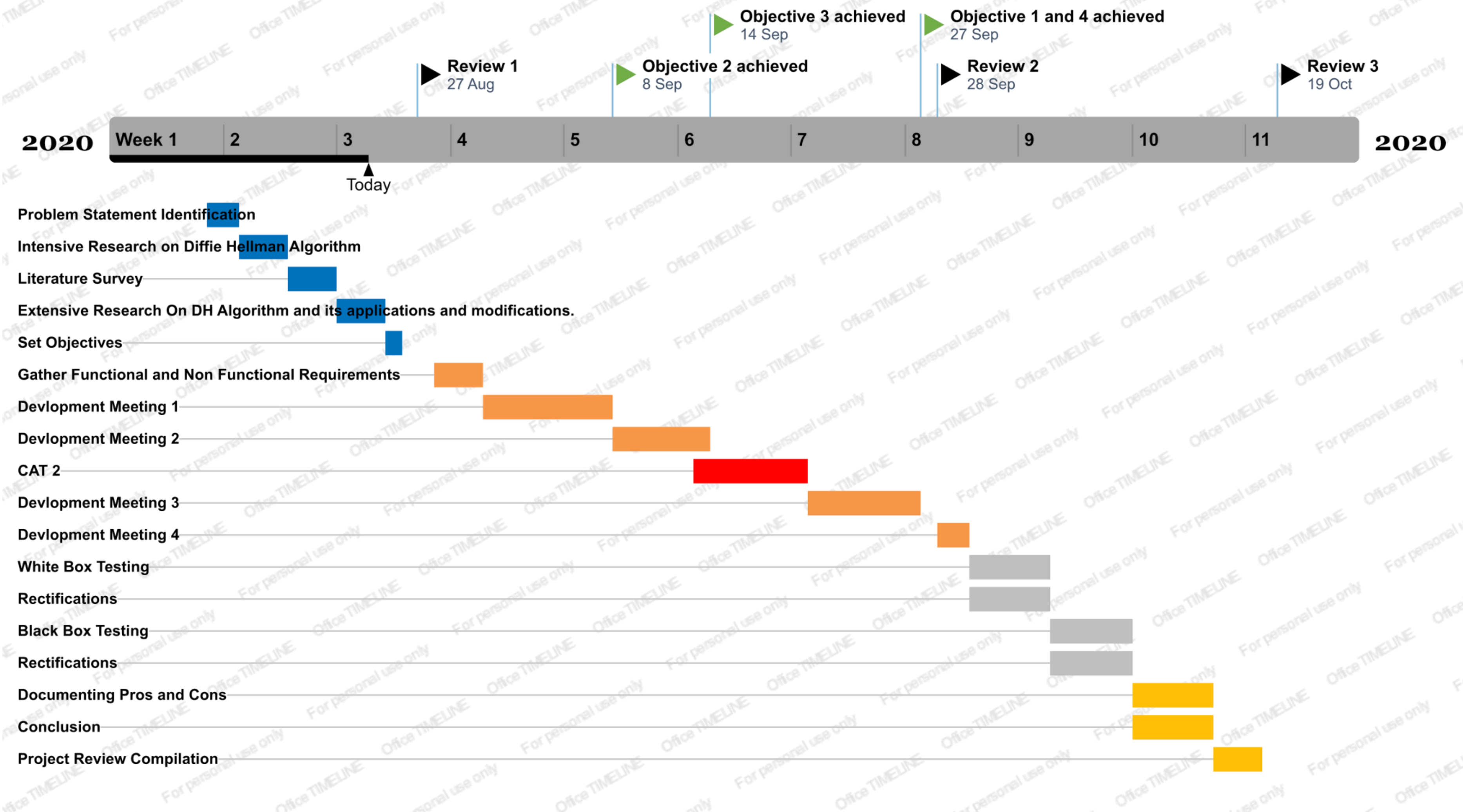
[18]	A Novel Multi-party Key Exchange Protocol	Protocol for multi party key sharing or key exchange that can be also used to share messages, among entities.	In this paper we see 3 algorithms/protocols which extend and improve the functioning of Diffie-Hellman Key Exchange Protocol. 1) Key exchange protocol 2) Protocol for encryption and decryption of the messages 3) Protocol using multiple bases we saw a optimized key sharing algorithm ,(Multi party key sharing key algorithm),that requires Euler's Totient function to operate.This particular Algorithm also employes massage sharing with minute variations. approach can be used for message sharing as well, in the above protocolthere were no restrictions on the value of'r'unlike that of'p'where it was required thatp is such thatp 1m odð/ðqÞÞexists Since'A'is generating the session key value it has high control over the values it can generate since there exists no dependency on the other entity unlike that of Diffie-Hellman key exchange where the resulting value depends totally on the basis of their (entities)	It is assumed during the implementation and design of the algorithm that heentity A or B wishes to communicate with other entity using some protocol'p'where entities are ensured as of whom they are communicating with.	The protocol will allow the entities to use multiple generators 'x', 'y' and this protocol has major advantage when compared with many multi-party key exchange protocols in terms of number of **iterations**, and also this protocol can be used for **sharing message** as well. The main advantage of this protocol is that both the entities are free to generate their own session key, and not be dependent on other entity to generate a session key for them. The ratio of the proposed algorithm with Diffie-Hellman algorithm in terms of iterations is half. The main initiating party is in possession of the session key and can eliminate the weak keys by choosing the secret	The major limitation is possibility of Generation of weak keys which have to be specifically dealt with. But they put the whole Algorithm's reliability on risk.

			secret values. moreover the credibility of the entire protocol lies on the assumption that it is difficult to solve the discrete logarithm problem.		number.	

[19]	Vulnerability Analysis And Security System For NFC-Enabled Mobile Phones using Diffie Hellman.	NFC is rooted in radio-frequency identification techn ology known as RFID which allows compatible hardware to both supply power to and communicate with an otherwise unpowered and passive electronic tag using radio waves. This is used for identification authentication and tracking.	In this paper, we use timestamp based Protocol where the size of the file and time at which it is sent is retrieved to calculate the estimated time of reception which will be compared with actual reception time is also proposed to prevent relay attacks. A complete study of authentication methodologies in NFC technology and a critical analysis of our proposed system with the existing systems showing how it can resist all the possible attacks on the data exchanged between mobile phones through NFC technology.	1. Size, Orientation, Angle, Placement are the various factors that affect an RFID technology. 2. The frequency range of an RFID system has a significant impact on the read range.	1. Our proposed solution attempts to resolve the critical attacks possible during data transfer through NFC technology. 2. Since it is a new technology, to the best of our knowledge there are not many attempts to resolve the security issues concerned with it.	1. It can only works in shorter distances which is about 10-20 cm. 2. It offers very low data transfer rates which is about 106 or 212 or 424 Kbps. 3. It is very expensive for the companies to adopt the NFC enabled devices.
[20]	Authenticated Diffie-Hellman Key Agreement Scheme that Protects Client Anonymity and Achieves Half-Forward Secrecy	The algorithm is proposed is for group communications. This algorithm proposes a way to generate a common key for group meeting which changes with change in membership of meeting. That means a new key is	This paper proposes a new problem: the modified elliptic curves computational Diffie-Hellman problem (MECDHP) and proves that the MECDHP is as hard as the conventional elliptic curves computational Diffie-Hellman problem (ECDHP). Based on the MECDHP, we propose 2 an authenticated D-H key agreement scheme which greatly improves client computational efficiency and protects client's anonymity from outsiders. This new	The parameters influencing the calculation is number of people involved. Since the DH calculation is already time consuming Other factors are dynamics of membership. As new members join or existing members leave the group the key	1. we have proposed a new authenticated D-H key agreement scheme which protects client anonymity and greatly improves client's computational efficiency 2. we have proposed a new authenticated D-H key agreement scheme which protects client anonymity and greatly improves	1 We cannot improve the computational efficiency of existent authenticated D-H key agreement schemes while still preserving the perfect forward secrecy

		generated when meeting starts and whenever someone leaves or when new or old member joins the meeting. This ensures that meeting channel is secure and safe and ensures both forward and backward secrecy	scheme is attractive to those applications where the clients need identity protection and lightweight computation	needs to be recomputed and hence performance is highly dependent.	client's computational efficiency	
[21]	Secure ECC-based RFID mutual authentication protocol for internet of things	The new protocol is developed by comparing it with other algorithms in terms of performance and vulnerabilities.	Implementing a light weight, backend independent, secure authentication protocol based on ECC in order to order to prevent attacks on wireless systems, IOT devices, devices using RFID authentication protocols and to make them light weight and optimal to that it could be implemented on light weight less storage and computation capacity devices. And to main and enhance availability, anonymity, mutual authentication and prevention of various prominent security attacks.	Choice of parameters in algorithm, order of elliptic curve, consideration of tag vulnerability and capacity to decide the weight of the algorithm.	Achieving various important security requirements all under one algorithm. Prevention of various attacks and compatibility with IOT devices which have limited storage and computational power	The algorithm covers various attack preventions but has not taken into the account of the vulnerabilities with some other non conventional attacks. Moreover, there is very less improvement in usage of storage space as compared with other algorithms in place

Network & Information Security Project Report



Proposed model / Algorithm / Framework

Computationally Modified Diffie-Hellman Key exchange algorithm working:

Client (Alice)	Server (Bob)
Prime number selected (p) is known to both client and server. Generator selected (g) is known to both client and server. Alice generates: Private keys: -> t (permanent key / long term key) -> x (ephemeral key) Public keys: -> T = (g^t) mod p (public counterpart of t) -> x (generated for each session and is private key of client) -> Calculate (x+t)%(p-1) Already has a nonce A,	Prime number selected (p) is known to both client and server. Generator selected (g) is known to both client and server.

Request 1 Client->Server :

Send
[A, (x+t) % (p-1), T] **---** T is required to send, only in case this is first
--- Request made since a new (t, T) is calculated

[A, (x+t)%(p-1)] **---** This can be sent in further requests.

Server Receives The Request

Case1:

[A , (x+t)%(p-1), T] is received
In this case server calculates modInverse(T,p) and updates the hash map with values **[A , modInverse(T, p)]**

Case2:
[A , (x+t)%(p-1)] is received
Server searches hash map for value of modInverse(T, p) of client identified by A and server finds it.

Case3:

[A , (x+t)%(p-1)] is received

Server searches has map for value of modInverse(T, p) of client identified by A and server couldn’t find any such value.

In *Case3* server raises an error and key-exchange is terminated. It can also send an acknowledgement back saying that proper parameters weren’t found.

In *Case1* and *Case2 following happens*

-> y is generated (private session key of server)

-> Y = g^y is calculated.

-> g^x is calculated

  g^x = (g^ ((x+t)%(p-1)) * modInverse(T, p).

-> K = (g^x)^y is calculated.

-> K = g^(x*y)

-> nounce B is known.

Request 2 Server-> Client:

Send
[B, Y]

Client receives [B, Y]
-> K = Y^x is calculated
-> K = (g^y)^x = g^(x*y)

K is our shared key know both to client and server

List of objectives achieved:-

- **Implemented** Computationally Modified Diffie-Hellman Key exchange algorithm in Java programming language with a Client Server architecture.

Results obtained:-

- **Reduced Computation Overhead on client side**
    1. The computation to calculate client’s public key is calculated on server side.
    2. On the server side, algorithm uses a hash map to store the T from the client for future use or generation of a session key in the future.
    3. So instead of multiple calculations server can fetch the public key in very short time.
    4. The client does not have to calculate T value again and again.

Remaining Work:-

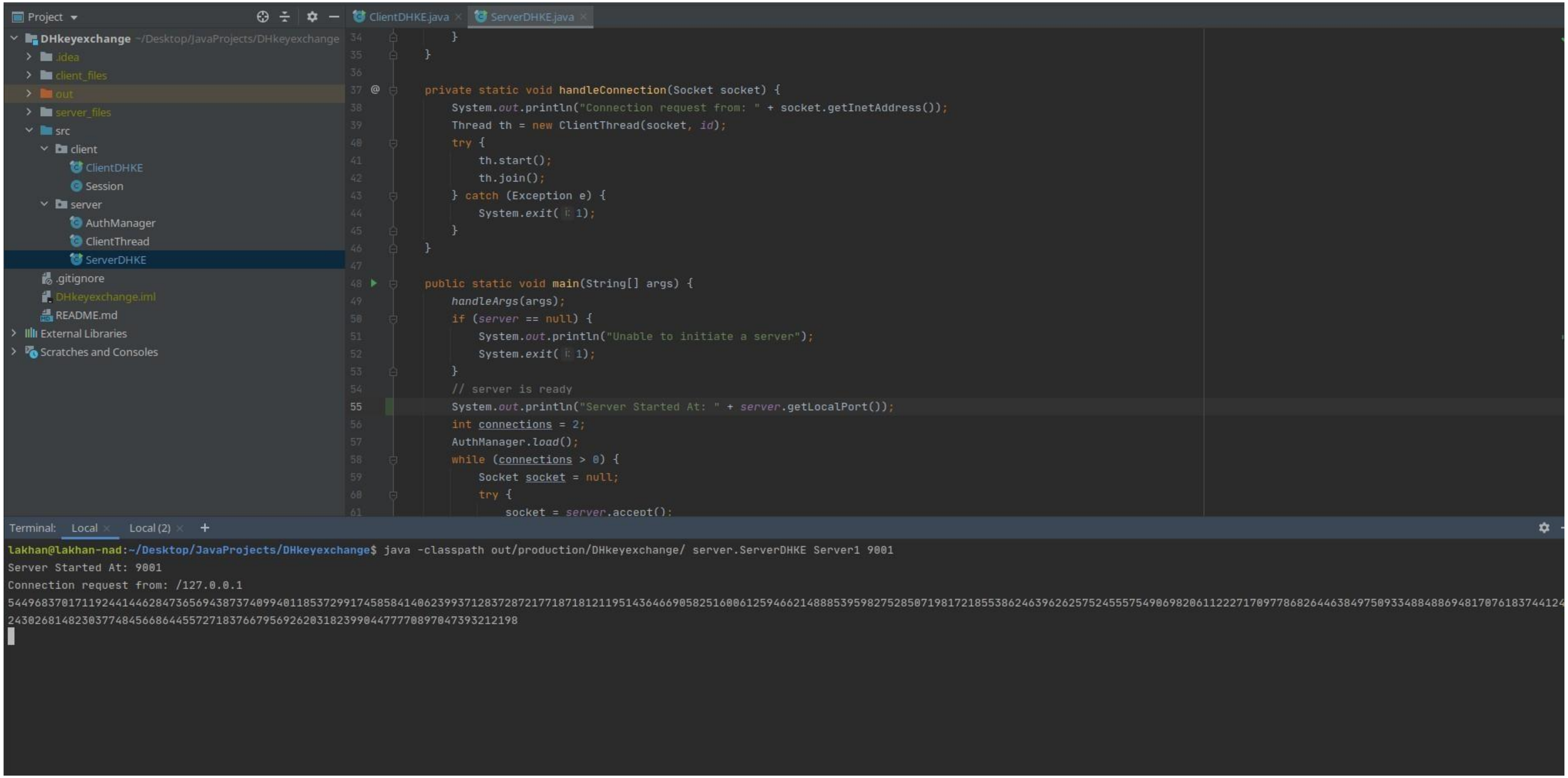
1. Comparative analysis of resources consumed by modified Diffie-Hellman key exchange algorithm and the regular Diffie-Hellman key exchange algorithm.
2. This is aimed to be achieved by system and process performance monitoring tool.

3. compare the time elapsed and memory consumed for different use cases of both modified Diffie-Hellman key exchange algorithm and Diffie Hellman Key exchange algorithm

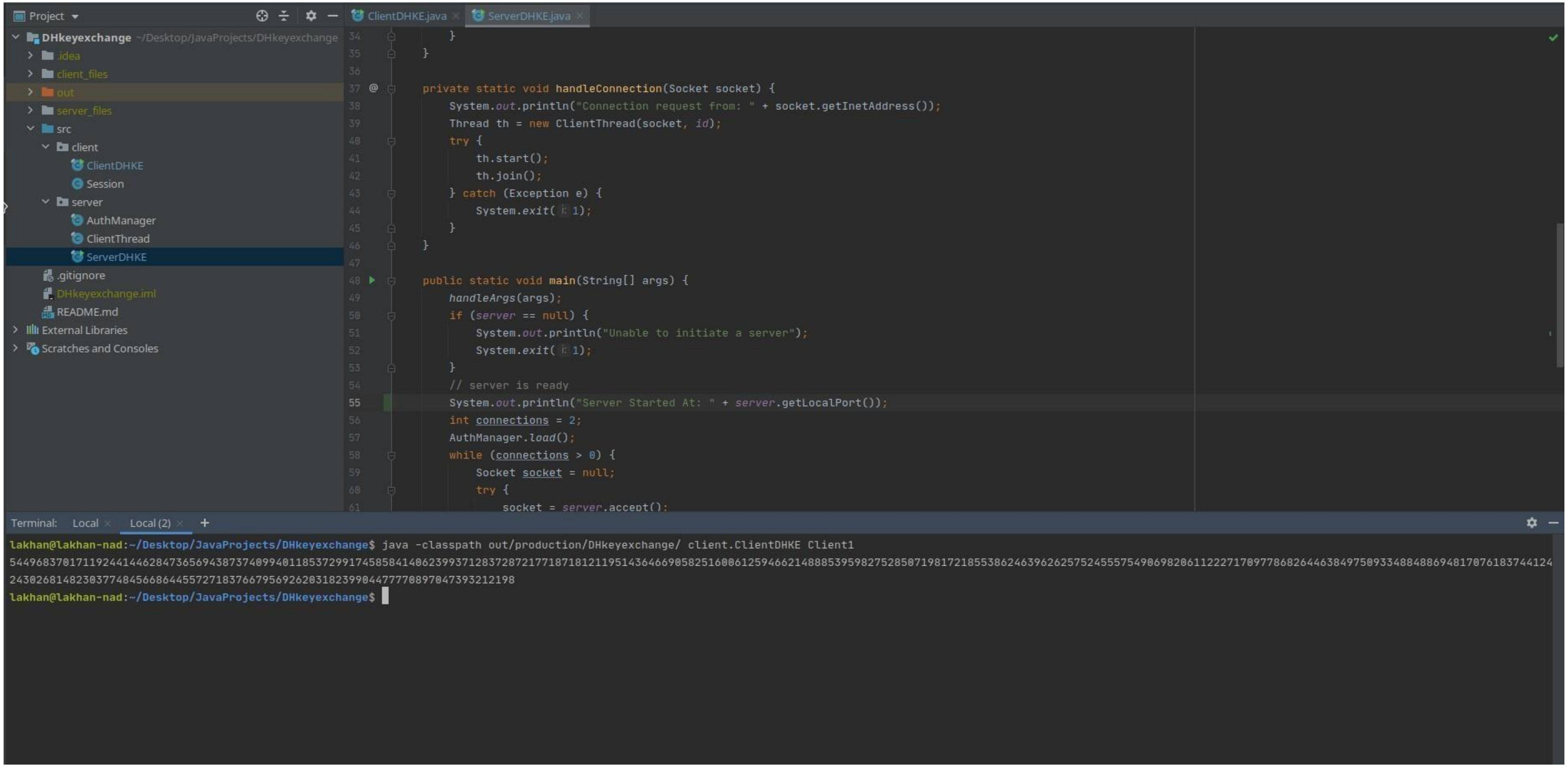
4. Analyse the effect of using a hash-map on the time elapsed while execution of the code and establishment of a shared key

<u>**IMPLEMENTATION:-**</u>

Server side



Client side


Project
DHkeyexchange ~/Desktop/JavaProjects/DHkeyexchange
.idea
client_files
out
server_files
src
client
ClientDHKE
Session
server
AuthManager
ClientThread
ServerDHKE
.gitignore
DHkeyexchange.iml
README.md
External Libraries
Scratches and Consoles
ClientDHKE.java
ServerDHKE.java
private static void handleConnection(Socket socket) {
System.out.println("Connection request from: " + socket.getInetAddress());
Thread th = new ClientThread(socket, id);
try {
th.start();
th.join();
} catch (Exception e) {
System.exit(1);
}
}
public static void main(String[] args) {
handleArgs(args);
if (server == null) {
System.out.println("Unable to initiate a server");
System.exit(1);
}
// server is ready
System.out.println("Server Started At: " + server.getLocalPort());
int connections = 2;
AuthManager.load();
while (connections > 0) {
Socket socket = null;
try {
socket = server.accept();
Terminal: Local Local (2)
lakhan@lakhan-nad:~/Desktop/JavaProjects/DHkeyexchange$ java -classpath out/production/DHkeyexchange/ client.ClientDHKE Client1
5449683701711924414462847365694387374099401185372991745858414062399371283728721771871812119514364669058251600612594662148885395982752850719817218553862463962625752455575490698206112227170977868264463849750933488488694817076183744124
2430268148230377484566864455727183766795692620318239904477770897047393212198
lakhan@lakhan-nad:~/Desktop/JavaProjects/DHkeyexchange$

Review-3

List of objectives given and achieved:

Found a solution for implementing a key exchange algorithm based on Diffie Hellman Key Exchange algorithm suitable for ioT ,embedded, and other resource constrained devices.

Designed caching technique for higher efficiency and lesser search times and search cycles ,

Implemented new connection request method supporting the above point.

Implemented a Modified Key exchange algorithm in Java.

Analyzed the space and time constraints that may and will arise.

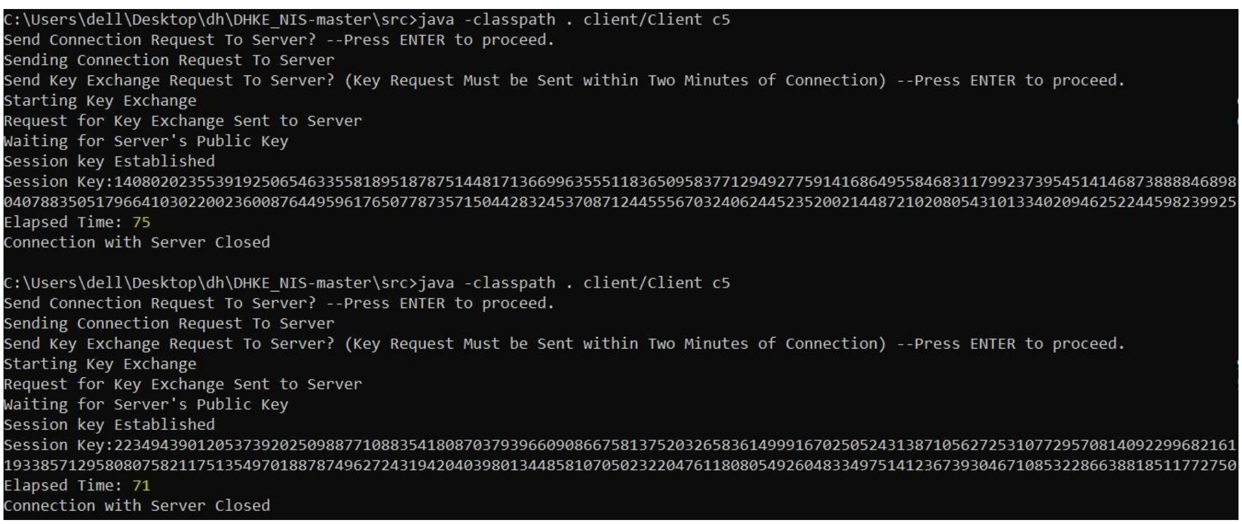
Tested the efficiency of caching and its result on overall algorithm time to establish a session and a shared key under different conditions.

Found a solution for implementing a key exchange algorithm based on Diffie Hellman Key Exchange algorithm suitable for ioT ,embedded, and other resource constrained devices.	The solution relies on research paper proposed by Hung-Yu Chien in his paper “A Generic Approach to Improving Diffie–Hellman Key Agreement Efficiency for Thin Clients”.
Designed caching technique for higher efficiency and lesser search times and search cycles ,	The server maintains a hashmap while running and a database while stop to store the modulo inverse of the public counterpart of a client’s permanent keys.
Implemented a new connection request method supporting the above point.	Used Sockets in Java to implement a client server model that before any communication makes a key exchange using the improved Diffie Hellman proposed above.
Implemented a Modified Key exchange algorithm in Java	The Language Java was chosen keeping in mind the fact that it is most popular choice for IOT devices and works on write once run anywhere motto.

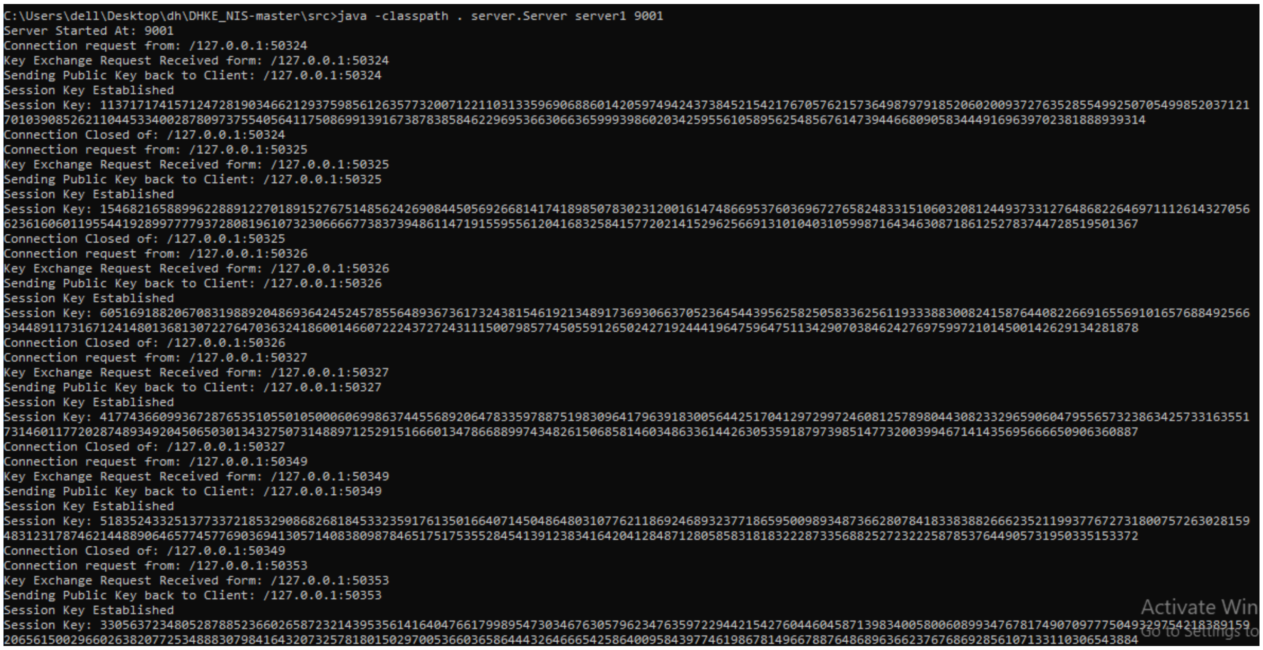
Analyzed the space and time constraints that may and will arise.	We ran the client and server daemons in a localhost environment to test the performance improvements of this algorithm from the existing Diffie Hellman Key Exchange Algorithm. And even in localhost the improvement was significant.
Tested the efficiency of caching and its result on overall algorithm time to establish a session and a shared key.	The efficiency of caching was tested with same client sending multiple key requests to same server and server using cached value of the client for further computations.

Experimental Results achieved:

Client-execution



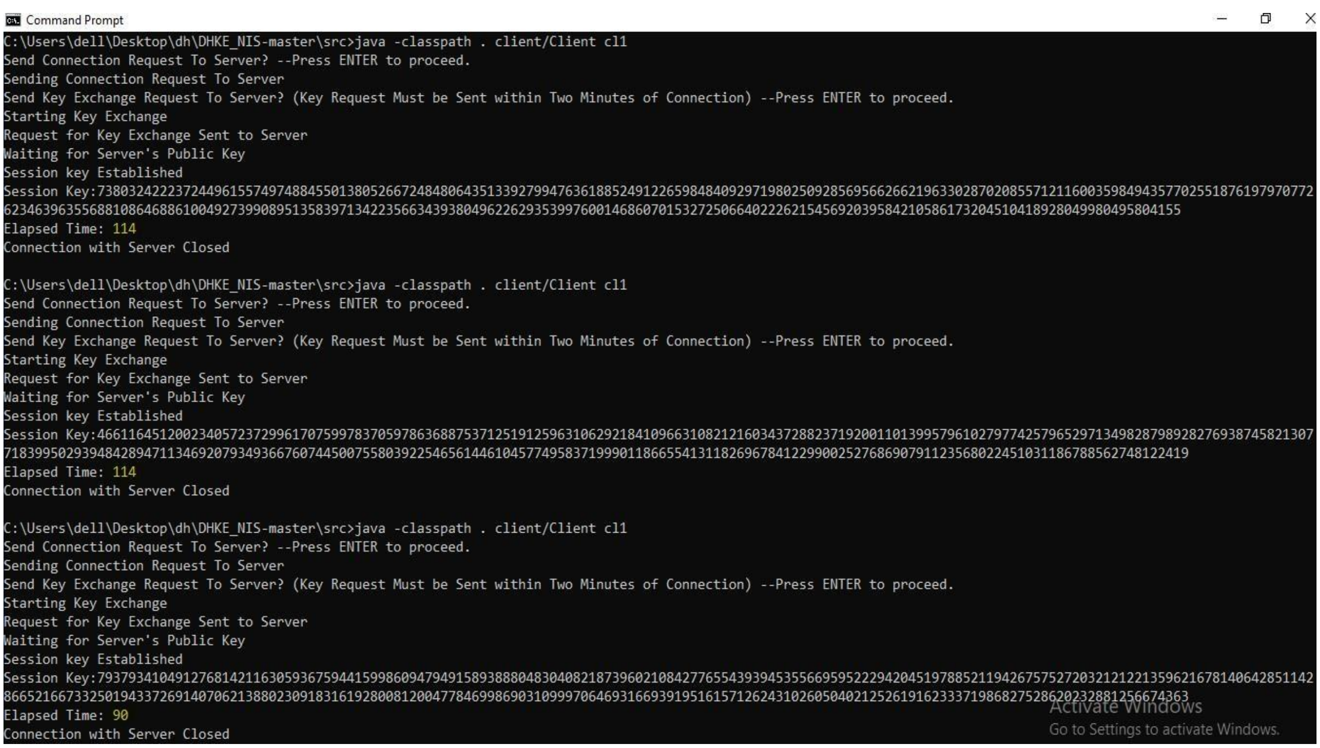
Server-execution



```
C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . server.Server server1 9001
Server Started At: 9001
Connection request from: /127.0.0.1:50324
Key Exchange Request Received form: /127.0.0.1:50324
Sending Public Key back to Client: /127.0.0.1:50324
Session Key Established
Session Key: 1137171741571247281903466212937598561263577320071221103133596906886014205974942437384521542176705762157364987979185206020093727635285549925070549985203712
1701039085262110445334002878097375540564117508699139167387838584622969536630663659993986020342595561058956254856761473944668090583444916963970238188893931 4
Connection Closed of: /127.0.0.1:50324
Connection request from: /127.0.0.1:50325
Key Exchange Request Received form: /127.0.0.1:50325
Sending Public Key back to Client: /127.0.0.1:50325
Session Key Established
Session Key: 154682165889962288912270189152767514856242690844505692668141741898507830231200161474866953760369672765824833151060320812449373312764868226469711126143270 56
6236160601195544192899777937280819610732306666773837394861147191559556120416832584157720214152962566913101040310599871643463087186125278374472851950136 7
Connection Closed of: /127.0.0.1:50325
Connection request from: /127.0.0.1:50326
Key Exchange Request Received form: /127.0.0.1:50326
Sending Public Key back to Client: /127.0.0.1:50326
Session Key Established
Session Key: 6051691882067083198892048693642452457855648936736173243815461921348917369306637052364544395625825058336256119333883008241587644082266916556910165768849256 6
9344891173167124148013681307227647036324186001466072224372724311150079857745055912650242719244419647596475113429070384624276975997210145001426291342818 78
Connection Closed of: /127.0.0.1:50326
Connection request from: /127.0.0.1:50327
Key Exchange Request Received form: /127.0.0.1:50327
Sending Public Key back to Client: /127.0.0.1:50327
Session Key Established
Session Key: 4177436609936728765351055010500060699863744556892064783359788751983096417963918300564425170412972997246081257898044308233296590604795565732386342573316355 1
7314601177202874893492045065030134327507314889712529151666013478668899743482615068581460348633614426305359187973985147732003994671414356956666509063608 87
Connection Closed of: /127.0.0.1:50327
Connection request from: /127.0.0.1:50349
Key Exchange Request Received form: /127.0.0.1:50349
Sending Public Key back to Client: /127.0.0.1:50349
Session Key Established
Session Key: 5183524332513773372185329086826818453323591761350166407145048648031077621186924689323771865950098934873662807841833838826662352119937767273180075726302815 9
4831231787462144889064657745776903694130571408380987846517517535528454139123834164204128487128058583181832228733568825272322258785376449057319503351533 72
Connection Closed of: /127.0.0.1:50349
Connection request from: /127.0.0.1:50353
Key Exchange Request Received form: /127.0.0.1:50353
Sending Public Key back to Client: /127.0.0.1:50353
Session Key Established
Session Key: 3305637234805287885236602658723214395356141640476617998954730346763057962347635972294421542760446045871398340058006089934767817490709777504932975421838915 9
2065615002966026382077253488830798416432073257818015029700536603658644432646665425864009584397746198678149667887648689636623767686928561071331103065438 84
```

Testing:

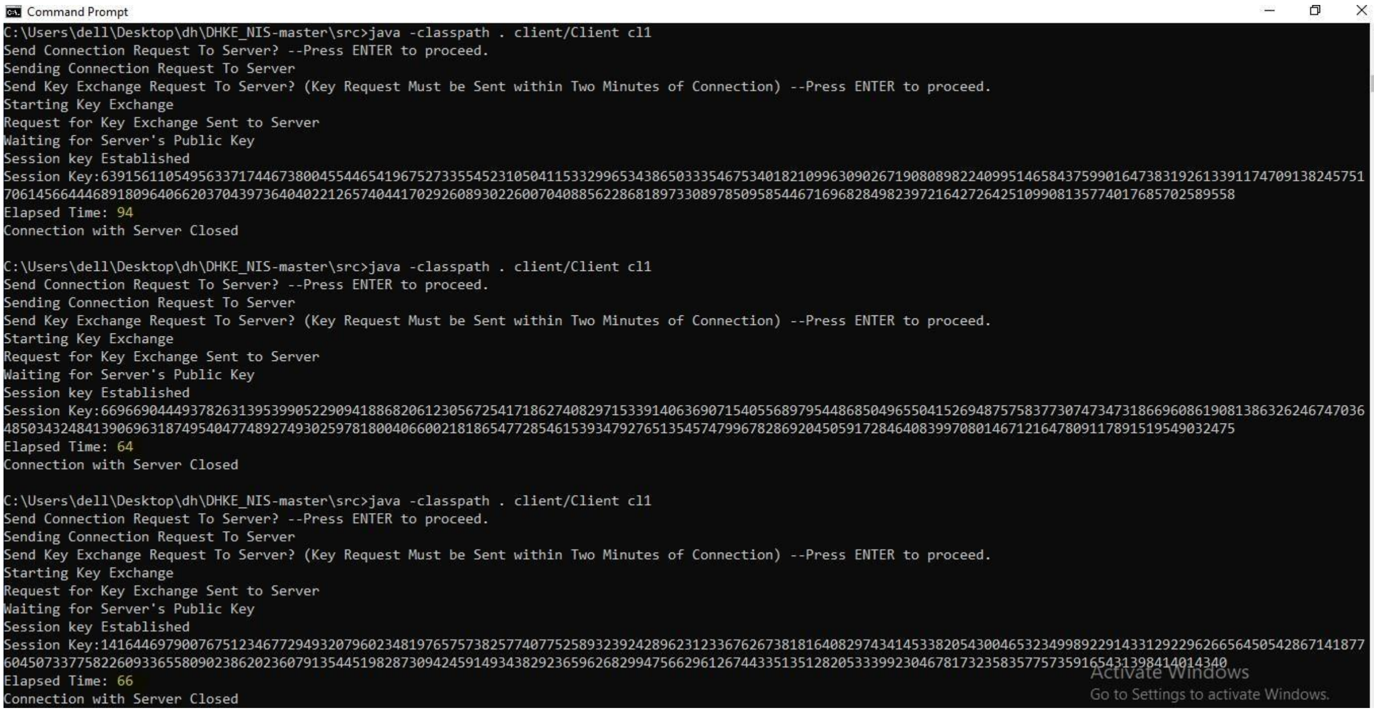
Diffie Hellman-Key Exchange Algorithm Implementation



```
Command Prompt
C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:73803242223724496155749748845501380526672484806435133927994763618852491226598484092971980250928569566266219633028702085571211600359849435770255187619797077 2
6234639635568810864688610049273990895135839713422356634393804962262935399760014686070153272506640222621545692039584210586173204510418928049980495804155
Elapsed Time: 114
Connection with Server Closed

C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:46611645120023405723729961707599783705978636887537125191259631062921841096631082121603437288237192001101399579610279774257965297134982879892827693874582130 7
7183995029394842894711346920793493667607445007558039225465614461045774958371999011866554131182696784122990025276869079112356802245103118678856274812241 9
Elapsed Time: 114
Connection with Server Closed

C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:79379341049127681421163059367594415998609479491589388804830408218739602108427765543939453556695952229420451978852119426757527203212122135962167814064285114 2
8665216673325019433726914070621388023091831619280081200477846998690310999706469316693919516157126243102605040212526191623337198682752862023288125667436 3
Elapsed Time: 90
Connection with Server Closed
```

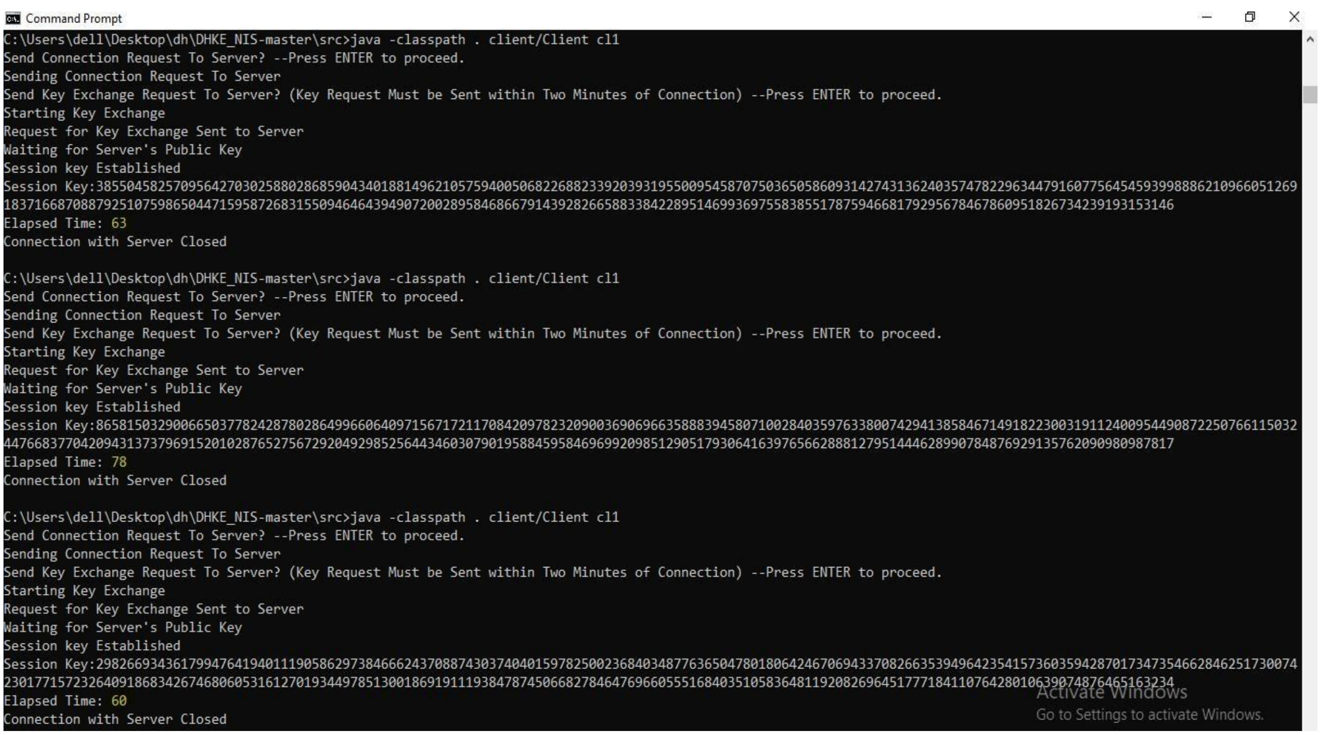


```
Command Prompt
C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:639156110549563371744673800455446541967527335545231050411533299653438650333546753401821099630902671908089822409951465843759901647383192613391174709138245751
70614566444689180964066203704397364040221265740441702926089302260070408856228681897330897850958544671696828498239721642726425109908135774017685702589558
Elapsed Time: 94
Connection with Server Closed

C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:669669044493782631395399052290941886820612305672541718627408297153391406369071540556897954486850496550415269487575837730747347318669608619081386326246747036
48503432484139069631874954047748927493025978180040660021818654772854615393479276513545747996782869204505917284640839970801467121647809117891519549032475
Elapsed Time: 64
Connection with Server Closed

C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:141644697900767512346772949320796023481976575738257740775258932392428962312336762673818164082974341453382054300465323499892291433129229626656450542867141877
6045073377582260933655809023862023607913544519828730942459149343829236596268299475662961267443351351282053339923046781732358357757359165431398414014340
Elapsed Time: 66
Connection with Server Closed
```

Activate Windows
Go to Settings to activate Windows.

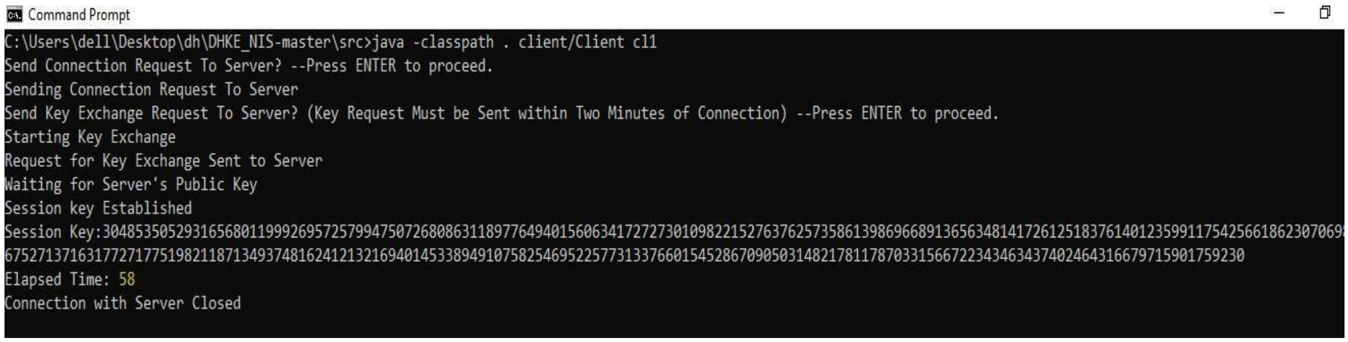


```
Command Prompt
C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:385504582570956427030258802868590434018814962105759400506822688233920393195500954587075036505860931427431362403574782296344791607756454593998886210966051269
18371668708879251075986504471595872683155094646439490720028958468667914392826658833842289514699369755838551787594668179295678467860951826734239193153146
Elapsed Time: 63
Connection with Server Closed

C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:865815032900665037782428780286499660640971567172117084209782320900369069663588839458071002840359763380074294138584671491822300319112400954490872250766115032
44766837704209431373796915201028765275672920492985256443460307901958845958469699209851290517930641639765662888127951444628990784876929135762090980987817
Elapsed Time: 78
Connection with Server Closed

C:\Users\dell\Desktop\dh\DHKE_NIS-master\src>java -classpath . client/Client cl1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Minutes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:298266934361799476419401119058629738466624370887430374040159782500236840348776365047801806424670694337082663539496423541573603594287017347354662846251730074
23017715723264091868342674680605316127019344978513001869191119384787450668278464769660555168403510583648119208269645177718411076428010639074876465163234
Elapsed Time: 60
Connection with Server Closed
```

Activate Windows
Go to Settings to activate Windows.



Validation:

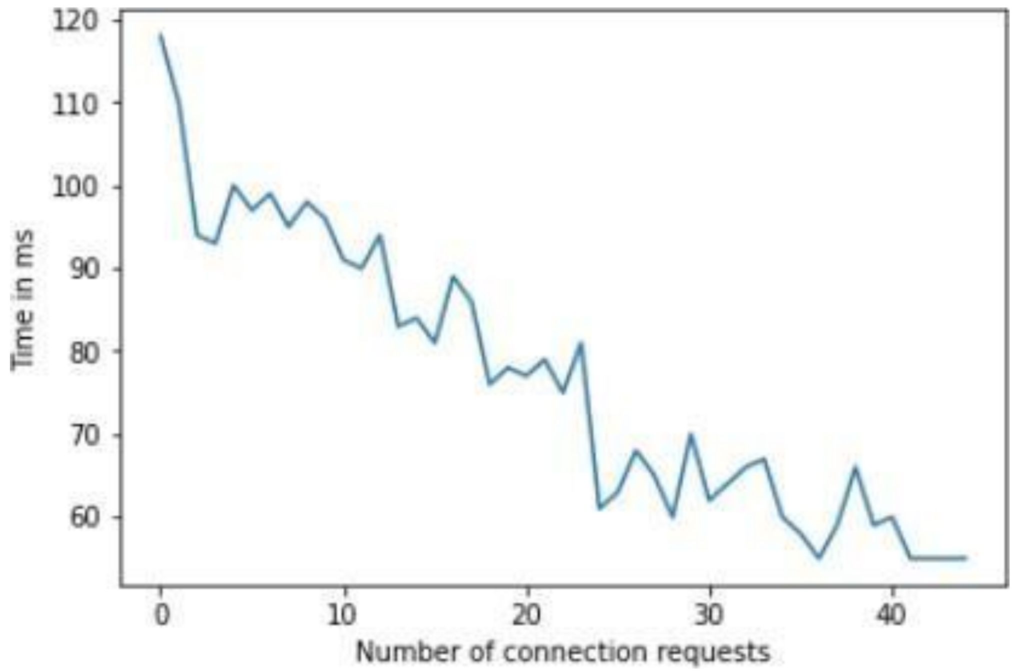
We can clearly see the reduced time elapsed

Generating the dataset:

Time elapsed in milli-seconds	Tth phase	Average of the interval
118 110	1	114ms
94 93 100 97 99 95 98 96 91 90	2	95.3 ms
94 83 84 81 89 86	3	86.1ms
76 78	4	78ms

77 79 75 81		
61 63 68 65 60 70 62 64 66 67 60 58 55 59 66 59 60 55 55 55 55	5	61ms

Overall analysis:

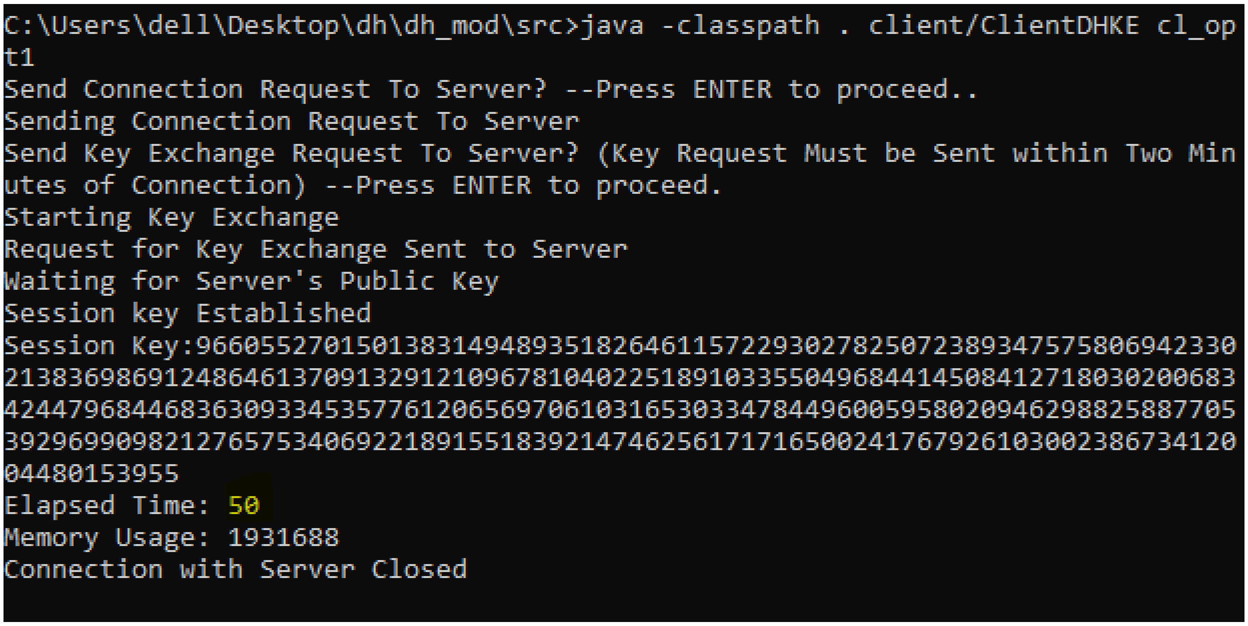


Modified diffie-hellman implementation:

```
C:\Users\dell\Desktop\dh\dh_mod\src>java -classpath . client/ClientDHKE cl_op
t1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Min
utes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:80386574245188856502446203043304071404875974388744094590294787763
0919883139613412851681530124482720051130769675469722964160165731711199357133 8
1249707496910970112278316965242271969094899185457708023159201155491741908318 7
7719340183286774892737622360766634385855383028715910313074780723285772100863 9
933054072482
Elapsed Time: 116
Memory Usage: 1939824
Connection with Server Closed
```

```
C:\Users\dell\Desktop\dh\dh_mod\src>java -classpath . client/ClientDHKE cl_op
t1
Send Connection Request To Server? --Press ENTER to proceed..
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Min
utes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:37793354811543204282192657225133695331146457585246721419637017973
53909272148676538517823434311116468669857364050034757720382799354952230063833
95078175400737026507131933866230952258855553993931305380239476537795936239061
16990871790225686828987511463508825141917934729713505810029195916806235101018
228299470003
Elapsed Time: 59
Memory Usage: 1940744
Connection with Server Closed
```

```
C:\Users\dell\Desktop\dh\dh_mod\src>java -classpath . client/ClientDHKE cl_op
t1
Send Connection Request To Server? --Press ENTER to proceed.
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Min
utes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:98424045805748037244817067589347992807341939316237574255701255521
94268765379554940544799879597566634684349211958910363601408513256830058791799
87072546339199437212714418654649171644632862844748990777536707641076183744699
61700032701763791184634039687893920745258239363101024943855579610253795366912
05439483934
Elapsed Time: 51
Memory Usage: 1931688
Connection with Server Closed
```

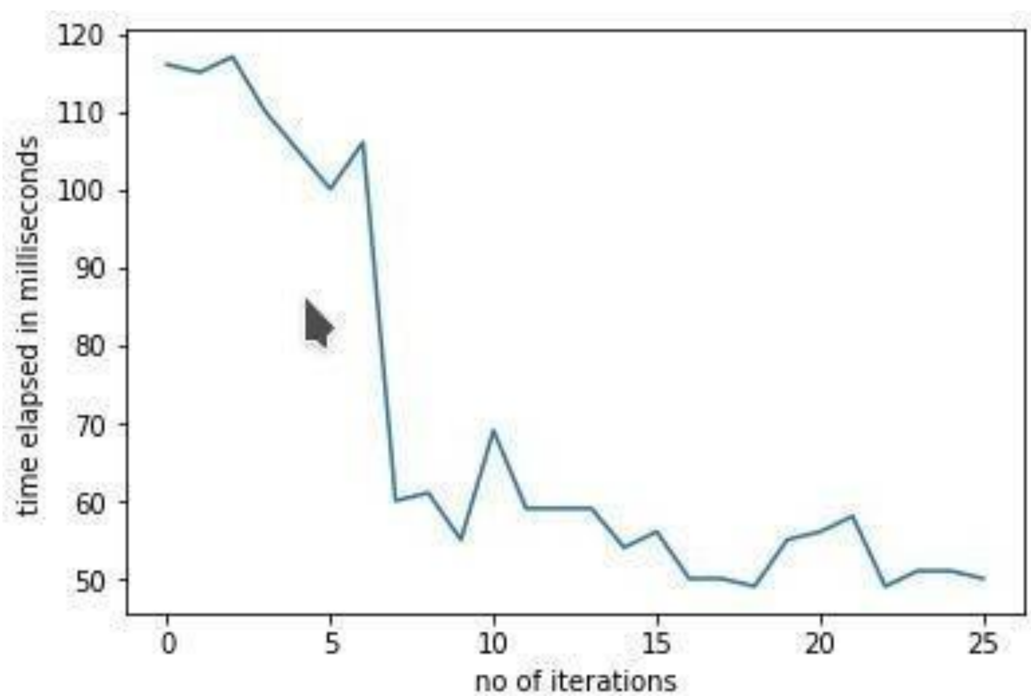


```
C:\Users\dell\Desktop\dh\dh_mod\src>java -classpath . client/ClientDHKE cl_op
t1
Send Connection Request To Server? --Press ENTER to proceed..
Sending Connection Request To Server
Send Key Exchange Request To Server? (Key Request Must be Sent within Two Min
utes of Connection) --Press ENTER to proceed.
Starting Key Exchange
Request for Key Exchange Sent to Server
Waiting for Server's Public Key
Session key Established
Session Key:9660552701501383149489351826461157229302782507238934757580694233O
2138369869124864613709132912109678104022518910335504968441450841271803020O683
42447968446836309334535776120656970610316530334784496005958020946298825887705
392969909821276575340692218915518392147462561717165002417679261030023867341 20
04480153955
Elapsed Time: 50
Memory Usage: 1931688
Connection with Server Closed
```

Time elapsed in milli-seconds	Tth phase	Average of the interval
116 **115** **117** **110** **105** **100** **106**	**1**	**109ms**
60 **61** **55** **69** **59** **59** **59** **54** **56**	**2**	**59.11ms**
50		**51.9ms**

50 **49** **55** **56** **58** **49** **51** **51** **50**	3	

Overall analysis:



Case of multiple request handling:

NORMAL DHKE

	Time(ms)	Memory(Bytes)
1 Key Request		
Client 1	132	1950520
Client 2	112	1950520
3 Key Request		
Client 1	216	1950520
Client 2	181	1950520
5 Key Request		
Client 1	304	2600504
Client 2	185	2600944
10 Key Request		
Client 1	354	2600632
Client 2	248	2600632

Modified Diffie Hellman

	Time(ms)	Memory(Bytes)
1 Key Request		
Client 1	82	1950500
Client 2	102	1950500
3 Key Request		
Client 1	195	1950464
Client 2	147	1950448
5 Key Request		
Client 1	249	2600188
Client 2	204	2600672
10 Key Request		
Client 1	314	2600784
Client 2	242	2600896

Inference:

The Buffering system using Hashmap resulted in a decrease of atleast 30% in elapsed times.

Along with shifting the Majority of the possible Arithmatic computations to the server side.

Diffie Hellman Key Exchange Algorithm

Phase:	Average reduction in time Elapsed wrt previous phase:	Average reduction in time Elapsed %
1	0%	0%
2	16.43%	16.43%
3	9.47%	24.56%
4	9.3%	31.57%
5	13.65%	46.49%

- The change can be explained in Existin Diffie Hellman approach as a result of the systems internal caching mechanisms and internal memory queing mechanisms

Modified Diffie-Hellman Key exchange Algorithm

Phase:	Average **reduction** in time Elapsed wrt previous phase:	Average **reduction** in time Elapsed %
1	0%	0%
2	45.8%	45.8%
3	11.86%	53.2%

- Here in this mechanism a reduced Elapsed time can be due to internal memory queing mechanisms and caching algorithms but a Significant drop in elapsed time can be observed.
- This can be explained due to the Hashmap which when a device contacts the server fior the first time it saves that clients's essentials,
- So when next time before calculations the essential values can can be quicky searched.
- This makes the reason for a steep reduction in elapsed time percentage.
- Note: we have tested the algorithm by repeatedly making connection requests to the client for the shared key to generate the test data and the following results
- .

Comparative study:

Comparison with Regular Diffie-Hellman key exchange algorithm:

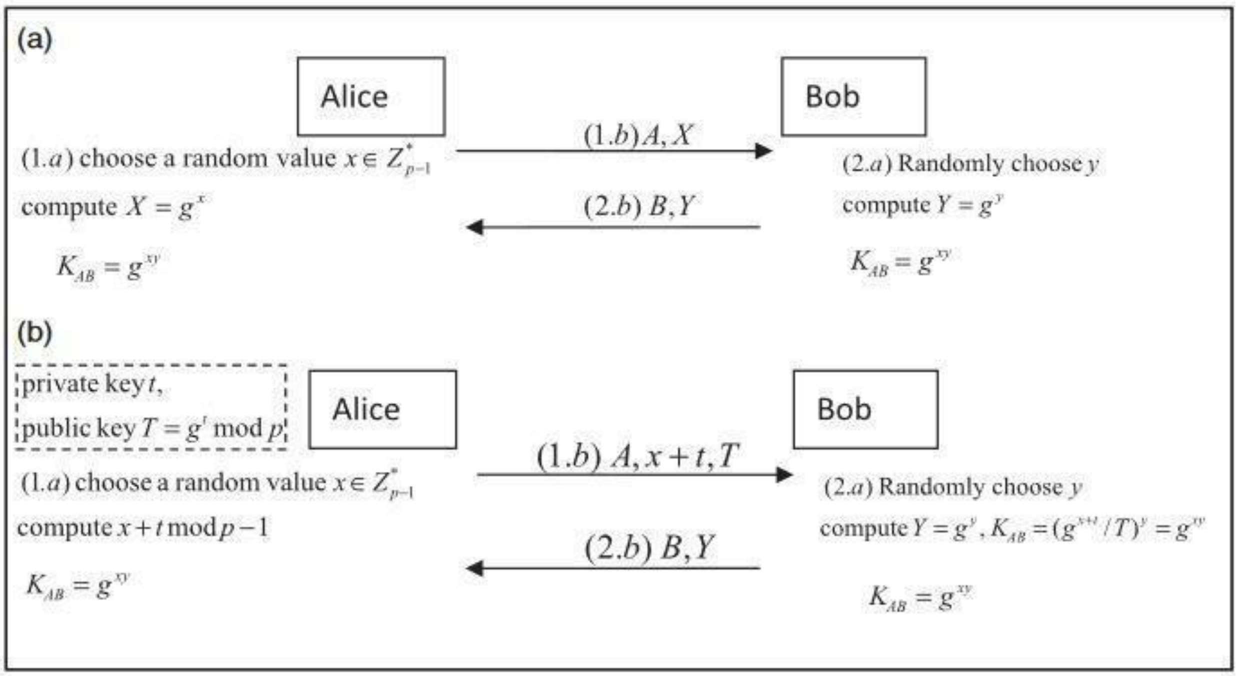


Figure .1

- Our major aim of the project was to implement the (b) part of the Figure.1 which is computationally modified Diffie-Hellman key exchange Algorithm for shifting the computations to the server side in order to minimize the computations at the client side.

- The part (a) in the given Figure.1 is the visual approach for Regular Diffie-Hellman Key Exchange Algorithm, and the Part b of the above given Figure 1(a) is the Visualization of Computationally modified Diffie-Hellman Key Exchange Algorithm.

Diffie Hellman :

- In this algorithm client and server model is not followed
- Both the parties are assumed to have enough computational resources and execute almost the same number of instructions
- Both parties have to calculate a random secret key
- Then calculate the public key
- Then send the public key to each other
- Then finally calculate the shared key
- We can see multiple multiplication cycles are involved at instruction level along with use of extensive modular arithmetic using large size prime integers
- This further adds up to the computation cost
- This algorithm is not feasible for the resource constrained devices.

Modified DH Algorithm :

- In our approach we work for the thin clients
- In multiple use cases thin clients ignore secure key exchange process on then fact that its computation is not possible during runtime.
- Our algorithm tries to transfer maximum possible computations to the server side
- It has a pre calculated private and public key saved
- And it also chooses a random key
- Now it sends nonce ,private key combined with random key and the public key to the server
- The server calculates the session key and then sends(server's) public key to the client(Thin client)
- Where the thin client calculates the session key with minimal computations.

Time and space Complexiety analysis:

Analyzing time and space

A constant time and space(memory) requirement for permanent key

Calculation of public key (T), raising generator to the power long term key, using modular function. This will be dependent upon the number of bits selected for permanent key, which will be further dependent upon the computational ability and availability device to be used

Calculation of empharel key for each session(time and space variant based on size)

Generating session key requires us to generate a random number, so we used random number generation fuction in java, complexity of which will not depend much upon the length of session key.

Java provides three ways to generate random numbers using some built-in methods and classes as listed below:

java.util.Random class: For using this class to generate random numbers, we have to first create an instance of this class and then invoke methods such as nextInt(), nextDouble(), nextLong() etc using that instance.

Math.random method : Can Generate Random Numbers of double type.

ThreadLocalRandom class: This class is introduced in java 1.7 to generate random numbers of type integers, doubles, booleans etc

Here we have generated random numbers of long data type using java.util.Random class. Which can be seen in session.java file. The time complexity of the random number generator is O(1). The time it takes does not increase as you have more random numbers.The randomness of java.util.Random could be an issue. It uses a seed of 2^48 so it will repeat itself after this many values. This means nextLong()

does not generate every possible value.If this is an issue you can use SecureRandom which is slower but the point it repeats is much higher.

Power function: another function which will be used multiple times for calculation of various important parameters for the session. We have used math.pow() function for this, which is briefed below:

The java.lang.Math.pow() is used to calculate a number raise to the power of some other number. This function accepts two parameters and returns the value of first parameter raised to the second parameter. There are some special cases as listed below:

- If the second parameter is positive or negative zero then the result will be 1.0.
- If the second parameter is 1.0 then the result will be same as that of the first parameter.
- If the second parameter is NaN then the result will also be NaN

Apart from that, we have used BigInteger data types in our code in order to have strong parameters, which cannot be figured out easily by attackers. But using this data type for calculations requires more space and time as compared to the primary data types. The main variation will come into picture in excecuting modular and exponential functions only, in order to have a more time and space efficient algorithm, various calculations are reduced on the client side provided its limited computational and storage abilities.

Result:

The following objectives have been achieved which were initially proposed to be completed:

1. To **find a solution** for implementing a improved Diffie-Hellman Key Agreement algorithm to improve the security of Embedded Systems and IoT devices with limited computational resources.

2. To **Implement** the computationally modified Diffie-Hellman Algorithm in Java.

3. **Testing the results** which include: Testing the Efficiency of the implemented algorithm along with comparison with existing diffie-Hellman Key exchange algorithm and obtain Quantitative and Qualitative measures to prove the feasibility of the given implementation

Some other results and key highlights:

1. **Designed** caching technique for higher efficiency and lesser search times and search cycles.*[discussed on pg. 27],*which resulted in decreased time for overall computations in the long term application of DH Algorithm which require repetitive connection and disconnection cycles.

2. **Implemented** new connection request method supporting the above point.[*Discussed on pg .28*]

Conclusion:

Our Study and concludes that the use of computationally modified Diffie Hellman Key exchange Algorithm allows the computationally constrained clients to use a secure key exchange mechanism as stated in [17] “ many thin clients do not use any security mechanism for saving computational resources”.

Our study provides the evidence of the given approach in [17] is **implementable** and also introduced 2 **new design methods** to make the implementation of Modified Diffie-Hellman key exchange mechanism to be easier, these methods as mentioned above in “Some other results and key highlights” section..

References:

[1] Nan Li, "Research on Diffie-Hellman key exchange protocol," 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, 2010, pp. V4-634-V4-637, doi:10.1109/ICCET.2010.5485276.

[2] Mishra, Manoj & Kar, Jayaprakash. (2017). A study on diffie-hellman key exchange protocols. International Journal of Pure and Apllied Mathematics. 114. 10.12732/ijpam.v114i2.2.

[3] G. Yang, J. Chen, Y. Lu and D. Ma, "An efficient improved group key agreement protocol based on Diffie-Hellman key exchange," 2010 2nd International Conference on Advanced Computer Control, Shenyang, 2010, pp. 303-306, doi: 10.1109/ICACC.2010.5486666.

[4] M. Abid and H. Afifi, "Secure E-Passport Protocol Using Elliptic Curve Diffie-Hellman Key Agreement Protocol," 2008 The Fourth International Conference on Information Assurance and Security, Naples, 2008, pp. 99-102, doi:10.1109/IAS.2008.22.

[5] Himanshi Chaudhary, Awadesh Kumar(2020) Sharma Hybrid Technique of Genetic Algorithm and Extended Diffie-Hellman Algorithm used for Intrusion Detection in Cloud 2020 International Conference on Electrical and Electronics Engineering (ICE3- 2020) Computing DOI: Computing DOI: 10.1109/CSNT.2013.97 (Pages)

[6] Mahmood Khalel Ibrahem(2012) Modification of Diffie–Hellman Key Exchange Algorithm for Zero Knowledge Proof 2012 International Conference on Future Communication Networks Computing DOI: 10.1109/ICFCN.2012.6206859(Pages)

[7] Chundong Wang, Chaoran Feng(2013) Security Analysis and Improvement for Kerberos Based on Dynamic Password and Diffile-Hellman Algorithm2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies Computing DOI: 10.1109/EIDWT.2013.49 (Pages)

[8] Mustafa Toyran, R Lavanya, Arjun Singh Group Key Exchange Protocol Based on Diffie-Hellman Technique in Ad-Hoc Network
Computing DOI: 10.1145/2659651.2659725(Pages)

[9] Prashant Rewagad ; Yogita Pawar (2013) Use of Digital Signature with Diffie Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing DOI: 10.1109/CSNT.2013.97(Pages)

[10] Mustafa Toyran ; Savaş Berber (2009) Efficient implementation of Diffie-Hellman (DH) key distribution algorithm in pool- based cryptographic systems (PBCSs)DOI: 10.1109/ELECO.2009.5355353 (Pages)

[11] Yun Chen; Xin Chen; Yi Mu (2006) A Parallel Key Generation Algorithm for Efficient Diffie-Hellman Key Agreement DOI: 10.1109/ICCIAS.2006.295289 (Pages)

[12] S V Sathyanarayan; R Lavanya (2017) Group diffie hellman key exchange algorithm based secure group communication DOI: 10.1109/ICATCCT.2017.8389149 (Pages)

[13] Ankit Taparia,Saroj Kumar Panigrahy,Sanjay Kumar Jena (22 February 2018)Secure Key Exchange Using Enhanced Diffie- Hellman Protocol Based on String Comparison DOI: 10.1109/WiSPNET.2017.8299856(Pages)

[14] Pratima Deshpande ; S Santhanalakshmi ; P Lakshmi ; Amrita Vishwa (Aug. 2017) Experimental study of Diffie-Hellman key exchange algorithm on embedded devices. DOI: 10.1109/ICECDS.2017.8389808(Pages)

[15] Arjun Singh Rawat ; Maroti Deshmukh (September, 2018) Efficient Extended Diffie-Hellman Key Exchange Protocol DOI: 8940607 (Pages)

[16] Lein Harn, Manish Mehta (March, 2004) Integrating Diffie–Hellman Key Exchange into the Digital Signature Algorithm DOI: 10.1109/LCOMM.2004.825705 (Pages)

[17] Hung-Yu Chien, A Generic Approach to ImprovingDiffie–Hellman Key AgreementEfficiency for Thin Clients, The Computer Journal Advance Access published November 2, 2015, from the British Computer Society, doi:10.1093/comjnl/bxv094

[18] Swapnil Paliwal and Ch. Aswani Kumar, A Novel Multi-party Key Exchange Protocol, Springer International Publishing AG, part of Springer Nature 2018A. Abraham et al. (Eds.): ISDA 2017, AISC 736, pp. 597–607, 2018.https://doi.org/10.1007/978- 3- 319-76348-4_58

[19]

[20] S.Kavya, K.Pavithra, Sujitha Rajaram, M.Vahini, N Harini, Vulnerability Analysis And Security System For NFC-Enabled Mobile Phones using Diffie Hellman, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 3, ISSUE 6, JUNE 2014, DOI:- 10.1109/EIDWT.2013.49

[21]Mustafa Toyran, R Lavanya, Arjun Singh Authenticated Diffie-Hellman Key Agreement Scheme that Protects Client Anonymity and Achieves Half-Forward Secrecy DOI: 10.1145/2659651.2659725(Pages)

[22] Amjad Ali Alamr · Firdous Kausar · Jongsung Kim · Changho, A secure ECC-based RFID mutual authentication protocol for internet of things, Springer Science+Business Media New York 2016, DOI 10.1007/s11227-016-1861-1