# CHAPTER 1: INTRODUCTION

With the rush of services and information provided on the Internet in present times, it has become extremely essential to authenticate the flow of information to the appropriate recipients. Since the volume of internet users and online data has gone beyond limits, it has become impossible for a single server-client model to cater the needs of today's internet users. Henceforth come the era of Multi Server–Client model in distributed network along with multi-server authentication protocol to check that the right information is being provided to the right user on internet.

A **user authentication scheme** is a mechanism that is used by a server to authenticate the user before he/she is allowed to access the service [13]. A considerable number of user authentication schemes have been proposed till now.

A **single–server user authentication scheme** is mechanism where a user is authenticated by a server prior to he/she is allowed to access services of that server. For a single server-client model, the remote machine access verification worked on the conventional authentication schemes which when applied to multi-server client model becomes impractical and inconvenient, since user has to memorize different pairs of identities and passwords to login to each one of the server.

A **multi–server user authentication scheme** is mechanism which is used by a set of multiple servers to authenticate the user before he/she is allowed to access the services of the respective set of multiple servers. Generally there are three kinds of participants in a multi-server user authentication system: users, a group of servers, and the authentication center. A user registered with the authentication center once can login to any server in the respective set of multi–server system i.e. they are not required to register themselves with different servers again and again within the multi–server system individually.

As the number of multi-server systems in the distributed environment increases, more user authentication schemes for multi–server environments have been proposed. A multi-server authentication schemes should fulfill the advantages of freely chosen password, no verification table, mutual authentication, single registration, secrecy and low computation and communication cost. However, a Multi–server authentication scheme is generally vulnerable to security attacks such as impersonation attack, password guessing attack, replay attack, insider attack, man-in-the-middle attack, forward secrecy, server spoofing and authentication center spoofing attack.

There are many authentication schemes developed for multi–server environment, but these all suffer from one or the other drawbacks.

So after exercising and analyzing these drawbacks, we are proposing "A Remote Authentication Methodology for Secure Communication in Distributed Network" which addresses the shortfalls of the existing multi–server environment.

## 1.1 Motivation

The motive behind designing this Remote Authentication Methodology for a distributed environment is to inherit the characteristic security features of Diffie-Hellman key exchange protocol and simultaneously overcomes the drawbacks of multi–server authentication schemes which provide a robust and invincible authentication mechanism. Generally, the security in distributed network is provided mostly through smart card based authentication schemes. The already proposed protocols in the literature are based on the idea of generation of keys before the session begins. However, due to the limitation on memory resources of smart card and openness of the internet environment, these proposed protocols are not able to achieve perfect security and also face a key management problem. These limitations and constraints laid down the foundation for the proposed work and became the key motivating factor for designing a secure and a scalable Mutual Authentication and Key Management Scheme in the domain of distributed network. Further the limitations and the constraints on memory resources and ease to generate and secure interface in the transfer of session key parameters inspired and encouraged us to use Diffie-Hellman key exchange protocol in domain of Distributed Networks.

## 1.2 Related Work

There are few works available in the field of mutual authentication and particularly for key generation in distributed network along with the use of Diffie-Hellman Cryptography within the same and consideration of the factors such as security, scalability, conformance, adaptability, reliability and less memory utilization plays a vital role in deciding the major characteristic features of any mutual authentication and key generation scheme for a distributed network. In the stated works various characteristic features have been taken into account along with the consideration of the protocol or scheme complexities involved in the processing, memory limitations, cost constraints. Some of which are stated and analysed below:

For single-server client model, lamport[1] proposed a conventional scheme which needed a verification table and it can be hacked easily. After this, many such schemes were proposed using one-way hash function without verification table. Some prominent schemes were given by hwang and liu[15], Sun[4], Das[16], Chen[17], Kim[18].

However such conventional schemes were inappropriate in terms of efficiency and convenience for users in the multi-server environment. Researchers then came up with multi-server authentication schemes. In 2001, Li[5] proposed such a scheme based on neural network. Lin used Euclidean plane and Elgamal digital signature for authentication in the paper[22]. Xie and Chen[11] presented another scheme using hash function and Xor operation. Using diffie-hellman, Zhu et al provide authenticity and session key generation in paper[12]. However, they all suffer from one or the other drawbacks, specified in the various papers [25] , [26], [10], [13], [16], [31], [20], [9], [32], [6], [12], [34], [11].

## 1.3 Problem Statement:

The main reason behind the proposed methodology in this thesis is to design and in turn successfully implement an efficient and a secure mutual authentication and key generation scheme using Diffie-Hellman key exchange protocol for Distributed Networks using appropriate simulation environment. The proposed scheme must satisfies the following set of security features: (1) No Verification Table (2) Freely Chosen Password (3) Mutual Authentication (4) Low computation and communication cost (5) Single registration (6) Session key agreement (7) User anonymity (8) Access Control (9) Security (10) Session Key Security (11) Known Key Security (12) Forward Secrecy[14]. Also the new authentication scheme should be successful to negate the following security attack on a given multi–server environment such as insider attack, impersonation attack, replay attack, password guessing attack, stolen–verifier attack and server spoofing attack[13]. The dissertation comprises of an analytical survey of the complexities involved in the various multi-server authentication schemes of distributed network platforms. This project attempts to address some of them with a summary of some early and current results from the implementation of proposed research. This thesis is to design, develop and propose:

**"A Remote Authentication Methodology for Secure Communication in Distributed Network"**

## 1.4 Scope of Work

The work done in this thesis is able to clearly demonstrate the importance of public key cryptography and its efficiency in mutual authentication and key generation scheme in domain of Distributed networks. Though the project follows a systematic, hierarchal, organised and a structured approach in proposing, demonstrating and implementing the vital statistics of the session key generation in distributed network but simultaneously it exploits the vital and beneficial characteristic features of an diffie-hellman in that scenario.

We have implemented the methodology on .Net framework using visual studio and C# language to demonstrate the working of the proposed methodology. The proposal consists of four phases: Server Registration, here each server registers itself at the authentication center; User Registration, the user registers with authentication center; Authentication of Remote User and Server, in this phase authentication center authenticates user and target server and vis-versa, after which, authentication center provides user and server a mutual key; Mutual Authentication and Session Key Generation, this is the last phase where the user and target server authenticates each other and generate session key.

The proposed work is confined to a single session establishment, mutual authentication, verification, acknowledgement and secure data exchange between the participating user and server at a particular instance of time. The proposed work can be very efficiently extended for a multisession establishment, mutual authentication, verification, acknowledgement and secure data exchange between two or more than two user and server terminals at a particular instance in multi-server environment.

The proposed scheme is independent of the local and global clock synchronous or asynchronous behaviour and is efficient and accurate in both the scenario.

The scope of the proposed scheme ranges from research domain to practical environment where the distributed systems play a vital role such as internet, research centers, retail markets, banking, industries, healthcare applications and its data security is of prime concern to us considering the processing limitations, memory constraints and security attacks on the usage of these.

## 1.5 Organization of Thesis

The remainder of this thesis is organized as follows:

Chapter 2 presents the historical work carried out in this area and the state of art in the multi-server authentication schemes. All the works taken from the literature and described in this chapter are related to the problem of efficiently utilizing the limited and constrained resources along with the maximum security defend with the possible use of Diffie–Hellman key exchange protocol, an asymmetric key cryptographic technique, in the domain of distributed environment for efficient and secure session establishment and secure communication through the correspondingly established channel.

Chapter 3 introduces the vital basic concepts behind successful working of the proposed scheme. It describes mathematical properties for a public key cryptographic system in the domain of Multi-Server Authentication scheme. From there it briefly explains the standard and widely accepted Diffie-Hellman key exchange protocol.

Chapter 4 gives the detailed description of the proposed scheme and the associated methodology employed in order to establish a secure communication channel for a particular session. First it introduces the various phases involved in the proposed scheme, followed by the terminology or notation used in describing the corresponding scheme for mutual authentication and key generation in the domain of Multi-Server environment. It then explains the proposed scheme in detail with related verifications of the corresponding phases. Finally it highlights architectural layout of the proposed scheme.

Chapter 5 presents the security perspective and performance analysis of the proposed scheme in a systematic and organized manner. It describes the detailed analysis of various security attacks on the proposed smart-card based multi-server authentication scheme in the distributed environment and gives implementation details and performance analysis of the corresponding scheme under distributed domain.

Chapter 6 describes the results of the proposed scheme. Several snapshots of the implementation results have been introduced in order to clearly demonstrate the working and efficiency of the proposed scheme.

Chapter 7 covers the conclusion and future prospective of the proposed work done in the thesis. It gives the conclusion remarks about the characteristic results achieved from the implementation of the proposed scheme and briefly highlights the enhancements which can be made to the current work for extending the proposal in future research.

Chapter 8 enlists the references used throughout the thesis and in the proposed scheme.

Appendix A enlists the source code of the various modules in the proposed scheme.

This chapter has discussed the overview of the entire thesis which in turn helps to analyze the thesis and to some extent describes what the proposed work is all about. Further Chapters will describe the proposed work in more detail and will give a rigorous analysis of the proposed scheme along with the facts of the respective implementation results or findings of this thesis along with the advantages and the limitations of the proposed scheme in the domain of Distributed network.

# CHAPTER 2: LITERATURE SURVEY

## 2.1 Objective:

Literature review constitutes an important section of my thesis. This chapter describes the various backgrounds that were considered during the thinking process. So following are the

objectives behind the literature survey conducted by me:

- Place each work in the context of its contribution to the understanding of the subject under review.
- Describe the relationship of each work to the others under consideration.
- Identify new ways to interpret, and shed light on any gaps in, previous research.
- Identify areas of prior scholarship to prevent duplication of effort.
- Point the way forward for further research.
- Place my original work in the context of existing literature.

## 2.2 State of Art

Security, scalability, computational and communication cost, efficiency, reliability and less memory utilization are major features of any key management algorithm or protocol of multi-server networks. In this literature survey we have considered the complexities, limitations, constraints of various authentication schemes of single-server and multi-server environment using smart card and also analyzed the Diffie-Hellman Key exchange protocol which is used for key exchange or sharing in that domain .

Basically this literature survey has analysis and rigorous coverage of mainly two domain specific research surveys and facts and features of both viz.

- ❖ Related Work of Cryptography
- ❖ Mathematical Overview
- ❖ Various Authentication Schemes for Distributed Network.

## 2.3 Related Work of Cryptography

### 2.3.1 Cryptography

Basic idea of cryptography is to mumble-jumble the original message into something that is unreadable or to something that is readable but makes no sense of what the original message is. To retrieve the original message again, we have to transform the mumble-jumbled message back into the original message again. So, **Cryptography** is the science of mathematics to “encrypt” and “decrypt” data. Cryptography enables us to store sensitive information or transmit it across insecure networks like Internet so that no one else other the intended recipient can read it. Cryptographic algorithms are mathematical functions that are used in the encryption and decryption process.

#### 2.3.1.1 Two Kinds of Cryptography Systems

There are two kinds of cryptosystems: symmetric and asymmetric. Symmetric cryptosystems use the same key (the secret key) to encrypt and decrypt a message, and asymmetric cryptosystems use one key (the public key) to encrypt a message and a different key (the private key) to decrypt it. Symmetric cryptosystems are also called as private key cryptosystems and asymmetric cryptosystems are also called as public key cryptosystems.

### 2.3.1.2 Symmetric Key Cryptography – An Overview

In symmetric-key cryptography, the sender and recipient agree beforehand on a secret private key. The plaintext is somehow combined with the key to create the cipher text. The method of combination is such that, it is hoped, an adversary could not determine the meaning of the message without decrypting the message, for which he needs the key. The following diagram illustrates the encryption process:

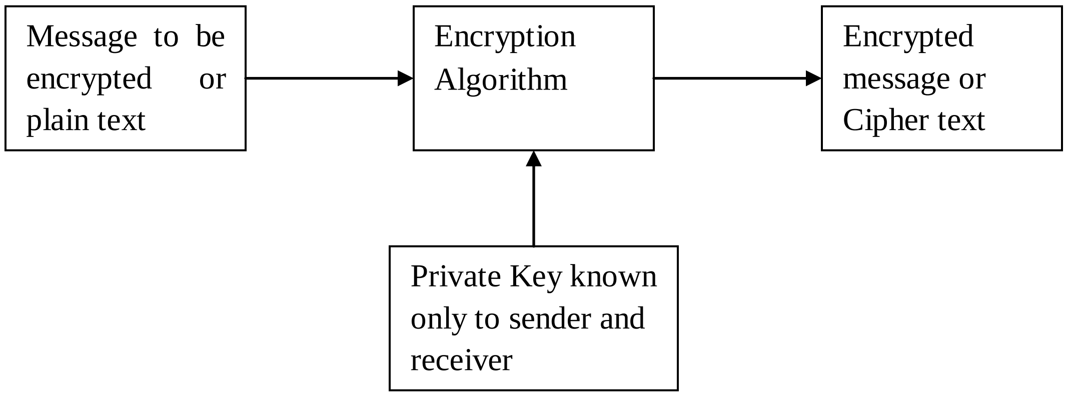

Message to be encrypted or plain text
Encryption Algorithm
Encrypted message or Cipher text
Private Key known only to sender and receiver

Figure 2.1: Symmetric Key Cryptography - Encryption Process

The following diagram illustrates the decryption process:

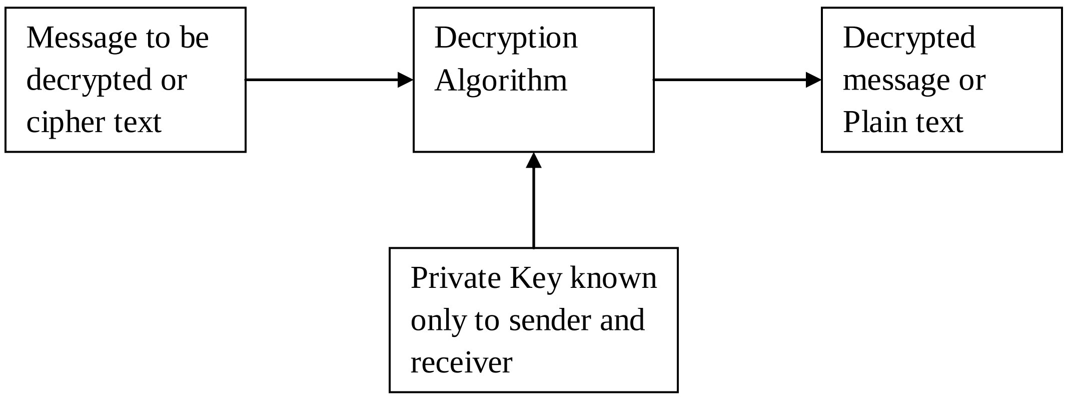

Message to be decrypted or cipher text
Decryption Algorithm
Decrypted message or Plain text
Private Key known only to sender and receiver

Figure 2.2: Symmetric Key Cryptography - Decryption Process

To break a message encrypted with private-key cryptography, an adversary must either exploit a weakness in the encryption algorithm, or else try an exhaustive search of all possible keys (brute force method).

### 2.3.1.3 Asymmetric Key Cryptography – An Overview

Asymmetric Key cryptography uses two keys Private key (known only by the recipient) and a Public key (known to everybody). The public key is used to encrypt the message and then it is sent to the recipient who can decrypt the message using the private key. The message encrypted with the public key cannot be decrypted with any other key except for its corresponding private key. The following Diagram illustrates the encryption process in the public key cryptography

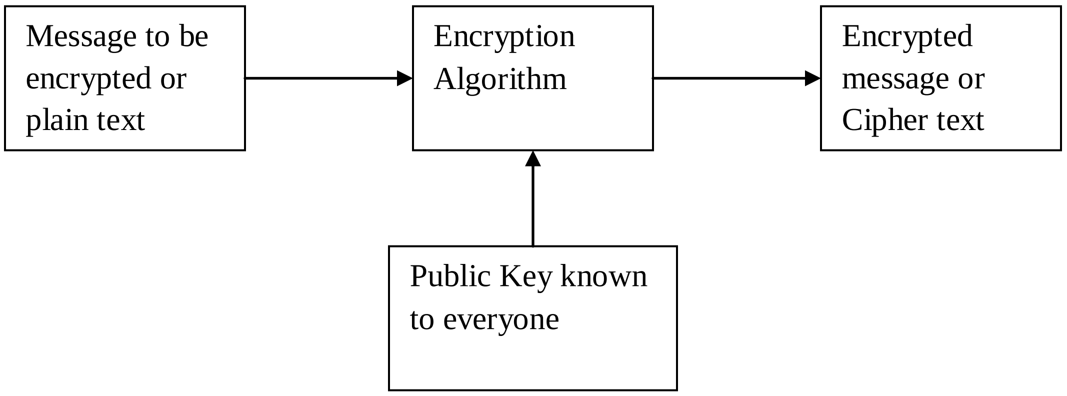

Message to be encrypted or plain text
Encryption Algorithm
Encrypted message or Cipher text
Public Key known to everyone

Figure 2.3: Asymmetric Key Cryptography - Encryption Process

The following diagram illustrates the decryption process in the public key cryptography:

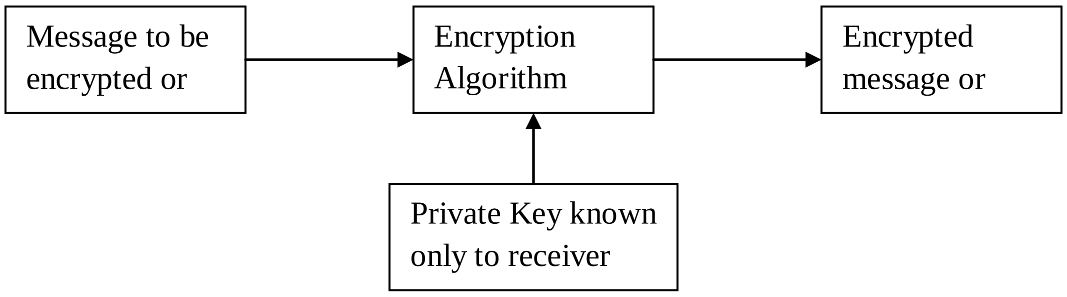

Message to be encrypted or
Encryption Algorithm
Encrypted message or
Private Key known only to receiver

Figure 2.4: Asymmetric Key Cryptography - Decryption Process

The public-key algorithm uses a one-way function to translate plaintext to cipher text. Then, without the private key, it is very difficult for anyone (including the sender) to reverse the process (i.e., translate the cipher text back to plaintext). Some examples of public-key cryptosystems are Elgamal (named for its inventor, Taher Elgamal), RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman), Diffie-Hellman (named, you guessed it, for its inventors), and DSA, the Digital Signature Algorithm (invented by David Kravitz).

### 2.3.2 Diffie-Hellman Key Exchange Protocol

Diffie-Hellman key exchange, also called exponential key exchange, is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming.

The most serious limitation of Diffie-Hellman in its basic or "pure" form is the lack of authentication. Communications using Diffie-Hellman all by itself are vulnerable to man-in-the-middle attacks. Ideally, Diffie-Hellman should be used in conjunction with an authentication method such as digital signatures to verify the identities of the users over the public communications medium. So, we have used Diffie-Hellman for the same proposes in our scheme i.e., to verify the identity of the each participant in the session key generation mechanism.

## 2.4 Mathematical Overview

### 2.4.1 Groups

A mathematical structure consisting of a set G and a binary operator $*$ on G is a group if,

- $\forall a, b \in G$, if $c = a * b$, then $c \in G$ (Closure)
- $a * (b * c) = (a * b) * c$, $\forall a, b, c \in G$ (Associative)
- $\exists\, e \in G$, such that $\forall a \in G$, $a * e = e * a = a$ (Identity element)
- $\forall a \in G$, $\exists\, a' \in G$ such that, $a * a' = a' * a = e$. $a'$ is unique for each a and is called the inverse of a.

The group is represented as $\langle G, *\rangle$. Additionally, a group is said to be abelian if it also satisfies the commutative property, i.e., $\forall a, b \in G$, if, $a * b = b * a$.

### 2.4.2 Modular Arithmetic

In modular arithmetic, the outcome of ‘a mod n’ is always a non-negative integer less than ‘n’. In other words, the result of the modulo operation with modulus ‘n’ is always an integer between 0 and n-1. This set of integer {0,1,2,….n-1} created by the modulo operation is, in modular arithmetic, referred to as the set of least residues modulo n or $Z_n$.

$Z_n = \{0, 1, 2, \ldots n-1\}$. $Z_n$ is the set of integers from 0 to n-1.

$Z_n$* is a sub-set of $Z_n$ that contains all the numbers that is less than ‘n’ and are relatively prime to ‘n’.

$Z_n^* = \{x < n-1 \mid x$ and n are relatively prime$\}$.

### 2.4.3 Additive Inverse

In $Z_n$, two integers are additive inverses of each other if

$$a + b \equiv 0 (\bmod n)$$

Basically, in modular arithmetic, $Z_n$ is a set of integers where each integer has its additive inverse.

### 2.4.4 Multiplicative Inverse

In $Z_n$, two numbers a and b are the multiplicative inverse of each other if

$$a * b \equiv 1 (\bmod n)$$

Basically, in modular arithmetic, an integer 'a' has a multiplicative inverse in $Z_n$ if and only if gcd(n,a) = 1, i.e. a and n are relatively prime. Thus, $Z_n$ is a set of integers where only some integers have multiplicative inverse. But, $Z_n*$ is a set of integers where each integer has a multiplicative inverse.

### 2.4.5 Generator 'g'

An element g is called a generator of a group G = <$Z_p$*, X>, if every element in *G* can be expressed as the product of finitely many powers of g.
i.e, $G = \{ e, g^1, g^2, \ldots\ldots, g^{n-1} \}$, where $g^n = e$ and 'e' is the identity element of the group G.

### 2.4.6 Cyclic Group

If the entire group can be generated using the power of an element, then the group is called a cyclic group. if a group has a generator, then it is a Cyclic Group.

## 2.4.7 Euler's Totient Function'$\phi(n)$'

The number of elements in $Z_n$ that are relatively prime to n. Hence $\phi(n) = |Z_n^*|$. In particular, $\phi(p) = p - 1$.

## 2.4.8 Order of the Group

The order of the finite group is the number of elements in the group. In G = $<Z_p^*,X>$, the order of group is $\phi(p)$.

## 2.4.9 Order of the Element

In group G = $<Z_p^*,x>$, the order of the element 'a' is the smallest integer i such that $a^i$ mod n = e mod n, where 'e' is the identity element of the G.

## 2.4.10 Primitive Root

In the group G = $<Z_p^*,x>$, when the order of the element is same as the order of the group($\phi(p)$), then that element is referred as the primitive root of the group G.

Each generator is a primitive root and can be used to generate the whole set. And if group is cyclic, it will have primitive roots.

In other words, if g is a primitive root of the group G = $<Z_p^*,X>$, the set $Z_p^*$ can be generated as,

$$Z_p^* = \{ g^1, g^2, g^3, \ldots\ldots, g^{\phi(p)} \}$$

### 2.4.11 Diffie-Hellman Key Exchange Protocol

In Diffie-Hellman, p is a large prime number and g is a generator of group $Z_p$* with multiplication as its operation, <$Z_p$*, X>. Here, $Z_p$* is same as $Z_n$* except that p is a prime. $Z_p$* contains all integers from 1 to p-1. Since p is a prime number, all the members of the set $Z_p$* have multiplicative inverse.

To implement Diffie-Hellman, the two end users A and B, while communicating over an insecure channel, mutually agree on positive whole numbers *p* and *g*, such that *p* is a prime number and *g* is a generator of *p*. The generator *g* is a number that, when raised to positive whole-number powers less than *p*, never produces the same result for any two such whole numbers. The value of *p* may be large but the value of *g* is usually small.

Once A and B have agreed on *p* and *g*, they randomly choose positive whole-number keys privately, *a* and *b*, both are less than the prime-number modulus *p*. Neither user divulges their personal key to anyone; ideally they memorize these numbers and do not write them down or store them anywhere. Next, A and B compute public keys *a** and *b** based on their personal keys according to the formulas

$$a^* = g^a \bmod p$$

and

$$b^* = g^b \bmod p$$

The two users can share their public keys *a** and *b** over a communications medium assumed to be insecure, such as the Internet or a corporate wide area network (WAN). From these public keys, a number $x$ can be generated by either user on the basis of their own personal keys. A computes $x$ using the formula

$$x = (b^*)^a \bmod p$$

B computes $x$ using the formula

$$x = (a^*)^b \bmod p$$

The value of $x$ turns out to be the same according to either of the above two formulas. However, the personal keys $a$ and $b$, which are critical in the calculation of $x$, have not been transmitted over a public medium. Because it is a large and apparently random number, a potential hacker has almost no chance of correctly guessing $x$, even with the help of a powerful computer to conduct millions of trials. The two users can therefore, in theory, communicate privately over a public medium with an encryption method of their choice using the decryption key $x$.

## 2.5 Various Authentication Schemes for Distributed Network

In the era of internet various types of information are shared among distant users. So it is very much necessary that the identity of the users gets authenticated before allowing the sharing of information[6].

Firstly in 1981, **Lamport** [1] proposed a single sever authentication scheme for authenticating users in the distributed environment like internet, which is a highly insecure network. However, the scheme proposed by him required the server to keep a verification table for the remote users. So, the major drawback of this schema was that information can be hacked by the hackers and thus, can be its security parameters can be compromised easily.

To use the network services provided by servers, password and hash function based authentication using smart card is one of the simplest and most widely used strategies. However, there are two weaknesses in the hash-based function schemes. The one is that the server should store verification table to verify the validity of the users, but it will suffer from the stolen-verifier attack. The other is, timestamps are used to avoid replay attacks, but it needs time synchronization [11]. So, to strengthen the security and lowers the communication and computation cost, a number of single-server authentication schemes using one-way hash function without verification table were proposed [2][3][4].

**Hwang and liu**[15] proposed a efficient and secure solutions where there is no necessary to maintain password table to verify the authentication of the user. But, it involves high communication and computational cost.

To remedy this, **Sun** proposed the revised version to significantly reduce the communication and computational costs [4]. However, it was for static user which limited its application and increase chances of ID-theft problem.

In order to overcome the ID-theft problem, **Das** gave a dynamic ID-based remote user authentication scheme using smart cards [16].

**Chien and Chen** proposed a revised version of the Das's scheme to conquer the weakness of the protection of user's anonymity [17].

However, the **Kim et al's** effort [18] is remarkable in order to guarantee user privacy against a remote server and traceable anonymity authentication along the user protection against the outside attacks.

But, these single server authentication schemes had certain shortcomings. Since, the user accesses services from more than one server, remote user authentication schemes for multi-server architectures, rather than single server architecture is considered. And if multiple servers architecture is considered for these conventional schemes designed for single server environment, the user must register their identities and passwords at these servers individually, which results in inefficiency for the scheme and highly hectic for the user to register at different servers individually.

As a result, the single-server architecture authentication schemes become highly inconvenient and impractical. And, as the amount of servers increases in the multi-server environment, many multi-server user authentication schemes for these environments came up [19] [20] [7] [5] [8] [21] [22] [10] [23] [24]. In these multi-server schemes, the remote user only registers with the authentication center once and can obtain services from multiple servers without repeating registration to every single server [11].

In 2001,**Li et al**[5] proposed a simple password authentication scheme for multi-server architecture in which the password authentication system is a pattern classification system based on neural networks without any verification table. Using neural networks for the scheme was a good approach but to train and maintain neural networks would take a great deal of time. The scheme cannot resist password guessing attacks and insider attack[25]. Their scheme did not provide mechanism for mutual authentication and session key agreement[13]. The costs such as computation and communication costs are extremely high as the scheme is based on the neural networks.

In 2003, **Lin et al**[22] gave a multi-server protocol based on ElGamal digital signature and geometric transformations on an Euclidean plane. But, this protocol has weaknesses and broken by **Cao and Zhong** [26].

In 2004 and 2005, **Tsaur et al** [27,23] gave two protocol. But, both their scheme were based on Lagrange interpolating polynomial which is computationally intensive.

In 2006 and 2007, **Cao et al.** [28] and **Hu et al.** [29] proposed an authentication scheme for multi-server environment. Both of their schemes assume that all servers are trustworthy. Nevertheless, this assumption is not always true as stated in [10].

In 2008, **Lee et al.** [30] proposed an authenticated key agreement scheme for multi-server using mobile equipment. However, their scheme cannot add a server freely. Because when a server is added, all users who want to login to this new server have to re-register themselves at the registration center for getting a new smart card. This increases the registration center's card-issue cost.

**Lin** [21] extended the work of Li et al [20] and revised their scheme by removing its vulnerabilities. However, this new scheme did not provide mechanism for mutual authentication and session key agreement[13].

**Juang** [7] also proposed a multi server authentication scheme in which symmetric encryption techniques without verification table was used which not only solved the problem of repeat registration but also satisfies computation efficiency. This scheme removed the weakness of **Lin**[21] and **Lie**[20] and provide mutual authentication and session key agreement mechanism. This scheme suffered from insider attack also[16].

**Ku et al.** [31] showed that Jung's scheme was vulnerable to the insider attack and could not provide forward secrecy.

In 2005, **Chang and Kuo** [19] proposed another authenticated key agreement protocol based on the Chinese Remainder Theorem and a modulus table.

**Huang and Shiau** [20] showed that both Jung[7] and Chang-Kuo[19] lacked of explicit key authentication and were inefficient with respect to communication costs. And they came up with an improved authentication protocol based on the line of geometry. However, Huang and Shiau's scheme still lacks of forward secrecy, and each server must maintain a user table[13].

**Liao and Wang** [8] proposed a dynamic ID based remote user authentication scheme for a multi-server environment. They utilized a dynamic ID instead of a static ID to achieve user anonymity. They indicated that a threat to user privacy is caused because of static ID for authentication. Hence, the scheme is appropriate for specific applications such as e-commerce[13]. However, their scheme is not secure. **Hsiang et al** [9] found that the Liao's scheme suffered from inside attacks**.** Liao and Wang scheme is vulnerable to server-spoofing attack and impersonation attack[32].

In 2008, **Tsai**[10] proposed an efficient multi-server authentication scheme based on one-way hash function without a verification table. In their scheme, the server do not maintain the verification table [10]. The scheme is based on the nonce, so it does not suffer from the time synchronization problem. Tsai claimed that their proposed scheme can satisfy the properties including mutual authentication, preventing the replay attack, preventing the spoofing attack, no verification table, and session key agreement[33].

**Yoon and Yoo**[6] claimed that the Jung and Tsai scheme are vulnerable to privileged insider attacks.

**Chen et al.**[34] showed that Tsai’s scheme cannot resist the server spoofing attack, and proposed a novel protocol. They claimed that the new protocol is not only the most secure but also the most efficient in a multi-server environment.

**Xie and Chen** [11] claimed that Chen et al scheme[34] cannot resist off-line password guessing attack and proposed a new scheme. However, we found that Xie and Chen scheme[11] is not resisted to insider attacker, impersonation attack and fails forward security.

**Zhu et al**[12] claimed that both the scheme Liao and Wang[8] and Tsai[10] suffers from the server spoofing attack and the parallel session attack and proposed a new scheme. However, we pointed out their scheme cannot resist impersonation attack.

This Chapter has presented a history and background on the work already done in the field. This presents fairly in detail what all is already done and presents a faint idea of what all could be done.

# CHAPTER 3: METHODOLOGY

This chapter describes the proposed methodology of the thesis. This starts with the description of the framework, then, we discuss the complete model so as to give a detailed basis for our thesis.

## 3.1 System Framework

An Overview of the entire system framework is presented in Figure 3.1. The whole methodology is divided into four phases: the first phase is server registration, then User Registration phase, after which comes the Authentication of Remote User and Server phase and final phase is Mutual Authentication and Session Key Generation. In the first phase, each server first sends its ID and registers itself with the authentication center. The user who wants to obtain services from a registered server at the authentication center, register himself/herself with a chosen Id and chosen password at authentication center during the user registration phase. In the authentication of remote user and server phase, the user logs in at the authentication center. The authentication center verifies the authenticity of the user and the remote server with which the user wants to connect. If the authentication is successful, then last phase execute in which the authentication center allows the user to communicate with the remote server directly and authenticate each other and establish session key. The computed session key is used in the service transaction over an insecure channel. The completed task flow is elaborated in the following sections.

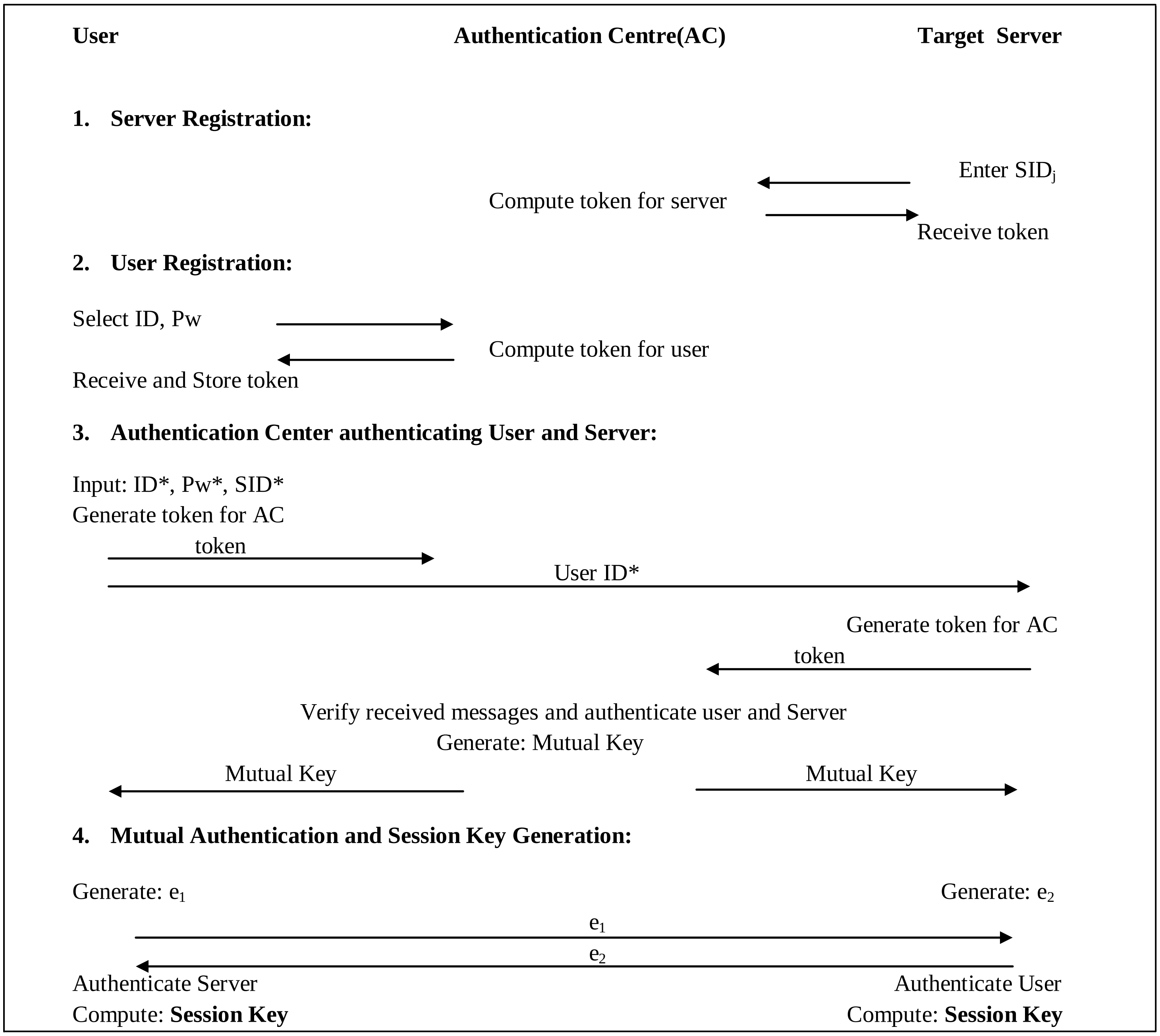


Figure 3.1: System Framework

## 3.2 Proposed Methodology

The proposed methodology takes ‘s’ servers, ‘n’ remote users and an authentication centre(AC). At the beginning, AC randomly chooses two secret numbers ‘x’ and ‘y’. When the user decides to login a server, he/she first registers themselves at the authentication center. The proposed methodology consists of 4 phases:

1. Server Registration,
2. User Registration,
3. Authentication of Remote User and Server,
4. Mutual Authentication and Session Key Generation.

The notations used in the scheme are showed in following table:

| Symbol | Definition |
|---|---|
| U, $S_j$ | The user and jth server, respectively |
| AC | The authentication center |
| ID, Pw | U’s identity and password, respectively |
| SID | $S_j$’s identity |
| x | AC generated random secret number for U |
| Y | AC generated random secret number for $S_j$ |
| K | User random nonce |
| h(.) | Secure one-way hash function |
| $\oplus$ | Bit-wise Exclusive-or(XOR) operation |
| $\|\|$ | Concatenation operation |
| A $\rightarrow$ B: M | A sends a message M to B |
| P | A large prime number |
| G | The primitive element in the GF(p) |
| a , b , c , d , a’ , b’ | Ephemeral random numbers in {1,……,p-1} generated by user, server and authentication center. |
| p, g | Publicly known Information |
| x, y | Information held by AC |

Table 3.1: NOTATIONS

As the proposed methodology is using Diffie-Hellman, ‘p’ and ‘g’ are two publicly known variables, where ‘p’ is a large prime number and ‘g’ is the generator of order p-1 in the group $<Z_p^*,x>$. The details for each phase are as follow,

## 3.2.1 Server Registration Phase:

In this phase, the server $S_j$ requests the authentication centre to register it in form of sending its ID, as shown in the figure 3.2. The following steps are performed during the registration phase.

- **$S_j \rightarrow$ AC: SID**

  In this, the server sends its identity ‘SID’ to the AC by a secure channel.

- **AC $\rightarrow$S: h(SID||y)**

  AC performs a hash operation on the concatenated value, obtained by the concatenation of the received server ID ‘SID’ and the secret value ‘y’, i.e.

  $$h(SID||y)$$

  AC sends back this computed value to server $S_j$ without storing it through a secure channel.

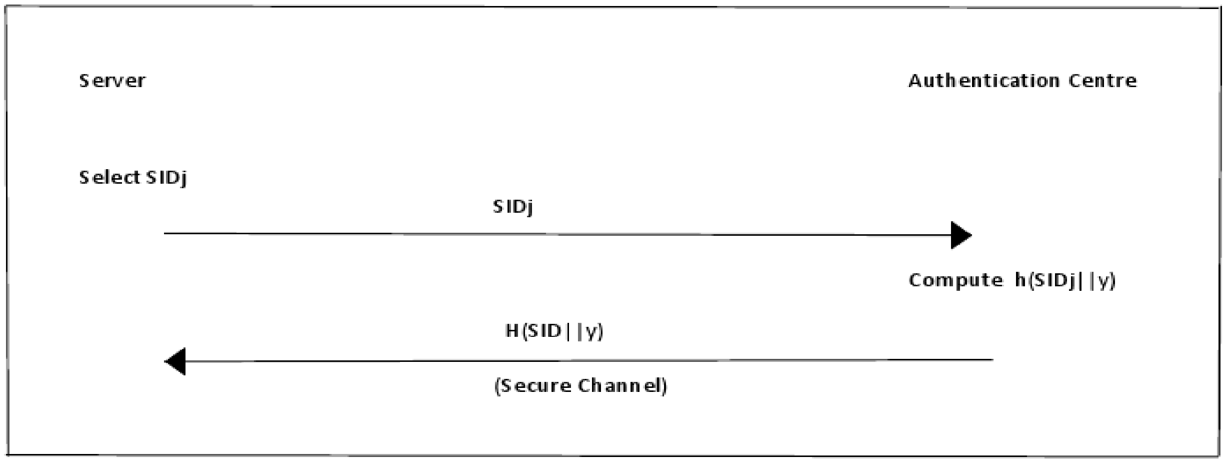


Figure 3.2: Server Registration Phase

## 3.2.2 User Registration Phase:

The user sends his identity 'ID' and XORed password '$Pw \oplus K$' to AC through a secure channel to get registered with the authentication center, as shown in the figure 3.3. AC performs following computation on them.

- **U → AC: ID, ($Pw \oplus K$)**

  After receiving the 'ID' and XORed password '$Pw \oplus K$' where K is a randomly chosen value at the user end only, AC computes '$R_u$' as a hash function on the concatenated value of 'ID' and the secret value 'x' and $C_0$ as an XORed value of $R_u$ and hashed value obtained on performing a hash function on the received value '$Pw \oplus K$'.

$$R_u = h(ID||x) \text{ and}$$

$$C_0 = R_u \oplus h(Pw \oplus K).$$

- **AC → U: $C_0$**

  AC sends $C_0$ back to user through a secure channel which gets stores on the user terminal or in the smart card but with no storage at the AC side.

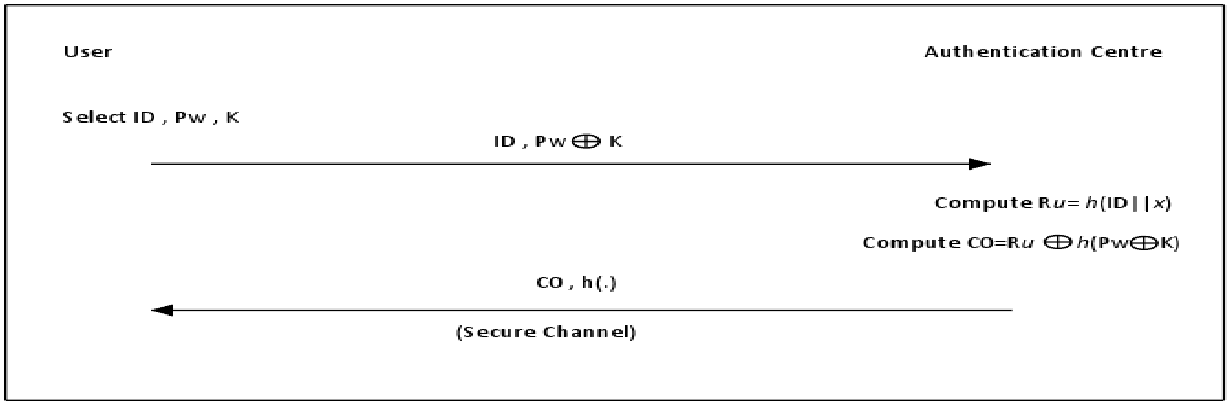


Figure 3.3: User Registration Phase

### 3.2.3 Authentication of Remote User and Server:

The following steps explain how a remote user generates a session key with the desired server through AC. The AC allows only the registered user to login and access the registered servers. The servers which are ready to provide their resources to user through AC, need to be logged in at the AC before this phase starts or else the user will not be able access that server through AC.As the scheme is maintaining no verification table either at AC end or at user or server end, the authentication center can still securely authenticate the validity of user and server and via-versa also, as shown in the Figure 3.4. In this phase, the communication is conducted between user and AC and between server and AC. The user and server do not communicate directly with each other. The AC does not generate any session key as that will be generated in the last phase. However, AC generates a mutual session key which is important in the session key generation. The steps are executed as follows,

**A.1 U → AC: ID*, SID*, $C_1$, $C_2$; U → $S_j$: ID***

When the user wants to access a server, registered on AC, the user enters his/her registered ID and password and the target server Id 'SID*' with which user desires to communicate. The user terminal computes $R_u$ using the stored value '$C_0$' in the smart card and the password entered by the user. 'K' is the randomly chosen value at the time of user registration with the AC. The AC obtained $R_u$ by XORing the '$C_0$' and hashed '$Pw \oplus K$'.

$$R_u = C_0 \oplus h(Pw \oplus K)$$

After retrieving $R_u$, the user randomly chooses 'a' ϵ $Z^*_p$ and computes $C_1$ as modulus operation p over the value, obtained by raising 'g' to power 'a' and $C_2$ as a hash operation on the concatenated value of $R_u$, SID* and $C_1$.

$$C_1 = (g^a)(\text{mod } p)$$

$$C_2 = h(R_u||SID^*||C_1)$$

On the computation of $C_1$ and $C_2$, user sends ID*, SID*, $C_1$, $C_2$ to AC and ID* to target server $S_j$ with which the user wants to communicate or of which the user wants to access services over the public network.

**A.2 $S_j \rightarrow$ AC: ID*, SID*, $C_3$, $C_4$**

On receiving the user request for accessing server 'j' resources, in the form of ID*, the server '$S_j$' extracts its stored hashed ID provided by the AC at the time of registration, it performs following operations on user id.

'Sj' randomly selects 'b' ϵ $Z^*_p$ and compute $C_3$ as modulus operation p over the value, obtained by raising 'g' to power 'b' and $C_4$ by performing hash operation on the concatenated value of h(SID||y), ID* and $C_3$. Here, h(SID||y) is the registered value with the AC stored at the server.

$$C_3 = (g^b)(\text{mod } p)$$

$$C_4 = h(h(SID||y)||ID^*||C_3)$$

After these computation, $S_j$ sends ID*, SID*, $C_3$, $C_4$ to the AC over the public network.

**A.3 AC $\rightarrow$U: $C_5$, $C_6$; AC $\rightarrow$ $S_j$: $C_7$, $C_8$**

On receiving messages in A.1 and A.2, the AC now verifies whether the received values, $C_2$ and $C_4$, over the public network are correct or not. For verification of received values, AC computes its own $C_{2'}$ and $C_{4'}$ by using received user id 'ID*', server id 'SID*', $C_1$, $C_3$, and its 'x' and 'y'. As AC does not maintain any verification table, AC calculates h(ID*||x) and h(SID*||y) on the received 'ID*' and 'SID*' for use in further operations. After calculating $C_{2'}$ and $C_{4'}$, AC checks them with the received $C_2$ and $C_4$ as follows,

$$h(h(ID^*||x)||SID^*||C_1) = C_2?$$

$$h(h(SID^*||y)||ID^*||C_3) = C_4?$$

If they both are equal, then AC authenticates the user with id 'ID*' and the server with id 'SID*' and provides them with further information. If either of them is not equal, AC does not authenticate that identity and rejects any further communication with that identity for that session.

After authenticating both the identities, AC chooses randomly 'c' $\epsilon$ $Z^*_p$ and 'd' $\epsilon$ $Z^*_p$ and computes $C_5$, $C_6$, $C_7$ and $C_8$. Here, the scheme computes two more variables $K_1$ and $K_2$ using $C_1$ and $C_3$ respectively.

$$C_5 = (g^c)(\text{mod } p)$$

$$K_1 = (C_1)^c(\text{mod } p) = (g^{ac})(\text{mod } p)$$

$$C_6 = h(K_1||h(ID^*||x)||SID^*)$$

$$C_7 = (g^d)(\text{mod } p)$$

$$K_2 = (C_3)^d(\text{mod } p) = (g^{bd})(\text{mod } p)$$

$$C_8 = h(K_2||h(SID^*||y)||ID^*)$$

After computing these values, the AC communicates $C_5$, $C_6$ to the user and $C_7$, $C_8$ to the server '$S_j$' on the public network.

## A.4 U $\rightarrow$ AC: $C_9$; $S_j$ $\rightarrow$ AC: $C_{10}$

On receiving the messages from AC, the user computes $K_1$ using modulus operation 'p' over the received value $C_5$ which is raised to power 'a', that is randomly chosen during calculation of $C_1$. After computing $K_1$, user computes $C_{6'}$ using this $K_1$, SID* and h(ID||x) as follows,

$$K_1 = (C_5)^a(\text{mod } p) = (g^{ac})(\text{mod } p)$$

$$C_{6'} = h(K_1||h(ID||x)||SID^*)$$

On calculating $C_{6'}$, user ensures its equality with the received value $C_6$ from the AC over the public network.

If it equals, the user authenticates the validity of AC and gets ensured about the security of network and accepts the communication with AC for further information and if $C_{6'}$ value comes out different then the received $C_6$, then User rejects any further communication with the AC and ends the session with it.

After the authentication of the AC, the user computes $C_9$ by performing a hash operation $K_1$. Here, the value of $K_1$ is updated by 1.

$$C_9 = h(K_1+1)$$

On the completion of the above operation, user sends $C_9$ to AC and asks for $C_{11}$ over the same public network.

Similarly on the target server end, the server '$S_j$' receives messages from AC and computes $K_2$ using modulus operation 'p' over the received value $C_7$ which is raised to power 'b', that is randomly chosen during calculation of $C_3$. After computing $K_1$, user computes $C_{6'}$ using this $K_2$, ID* and h(SID||y) as follows,

$$K_2 = (C_7)^b(\text{mod } p) = (g^{bd})(\text{mod } p)$$

$$C_{8'} = h(K_2||h(SID||y)||ID^*)$$

On calculating $C_{8'}$, server ensures its equality with the received value $C_8$ from the AC over the network.

If it equals, the server authenticates the validity of AC and gets ensured about the security of network and agrees for further communication with AC and if $C_{8'}$ value comes different, then server rejects any communication with AC and ends the session there only.
On the authentication of AC, the user computes $C_{10}$ by performing a hash operation over $K_2$ which is incremented by 1.

$$C_{10} = h(K_2+1)$$

After completion of the above operation, server sends $C_{10}$ and asks for $C_{11}$ from AC over same public network.

### A.5 AC → U: $C_{11}$; AC → $S_j$: $C_{11}$

When AC receives $C_9$ and $C_{10}$ from the user and server respectively, it verifies the validity of these received values for further authenticity of the user and server and ensures itself that the network is secure for transmitting mutual session key to user and server. It calculates $C_{9'}$ by incrementing its own $K_1$ value by 1 and performing a hash operation on it. Similarly, it gets C10' by performing hashing on the value obtained by updating $K_2$ by 1.

$$C_{9'} = h(K_1+1)$$

$$C_{10'} = h(K_2+1)$$

After these calculations, the AC checks their equality with $C_9$ and $C_{10}$, respectively. If they both are equal, AC authenticates the user and target server completely, as $K_1$ and $K_2$ values sent in $C_6$ and $C_8$ are received and sent back by the intended receivers correctly and safely. And either of them is not equal, AC stops immediately any communication with both the ends and ends the session, assuming that the user or the server or the network is not secure for communicating any information to the user or server.

After the final authentication of user and server, AC generates a mutual session key '$C_{11}$' for the user and target server. This mutual key is used by the user and target server for further communication between them. The key is obtained by XORing the information held by both user and server so that user can extract the server information and server can extract the user information using this key. It contains the hashed value of user information, i.e. '$h(ID^*||x)||SID^*||K_1+2$' and hashed value of server information, i.e. '$h(ID^*||x)||SID^*||K_1+2$'. The computation of $C_{11}$ is as follows,

$$C_{11} = h(h(ID^*||x)||SID^*||K_1+2) \oplus h(ID^*||x)||SID^*||K_1+2)$$

Once the $C_{11}$ is computed, it sent to user and server over the same public network from where AC received $C_9$ and $C_{10}$. The AC does not conduct any communication with them after this. User and target server is allowed to communicate with each other directly and generate the session key. This step marks the end of AC involvement.

### 3.2.4 Mutual Authentication and Session Key Generation Phase:

This phase is an important one as it generates the session key. In this phase, the user and target server communicate directly and mutually authenticate each other after which they involve in the process of session key generation. There is no communication with AC for any information. This phase ends with the generation of session key between user and target server. The generated session key is only known to the user and the target server. No one else have any information for the generation of session key, even AC has no information how the session key is generated. Section B of figure 3.4 shows the mutual authentication of the user and the target server. The details of each step are as follows.

**B.1 U → Sj: $C_{14}$**

On receiving the mutual session key '$C_{11}$' from AC, the user U chooses $a' \epsilon Z^*_p$ and extracts the server information '$C_{12}$' which is $h(h(SID||y)||ID^*||K_2+2)$, from $C_{11}$ by XORing $C_{11}$ and user information $h(h(ID||x)||SID^*||K_1+2)$. Here, the user increases value of $K_1$ by only 1 as it has already been incremented by 1 in $C_9$. The computation of $C_{12}$ is following,

$$C_{12} = C_{11} \oplus h(h(ID||x)||SID^*||K_1+2)$$

After the extraction of $C_{12}$, user computes two new values, '$C_{13}$' which is a modulus operation 'p' on the value obtained by raising 'g' to the power 'a'' which is a randomly chosen value form the group $<Z_p^*,x>$ and '$C_{14}$' which is a hashing operation on the XORed value of '$C_{13}$' and the extracted '$C_{12}$'.

$$C_{13} = (g^{a'})(\text{mod } p)$$

$$C_{14} = h(C_{13} \oplus C_{12})$$

However, the user transmits only $C_{14}$ to the server Sj through the public network and keeps the $C_{13}$ and $C_{12}$ for further computation.

**B.2 $S_j \rightarrow U$: $C_{17}$**

Similarly, when the target server '$S_j$' receives the mutual session key '$C_{11}$' from AC, $S_j$ extracts the user information '$C_{15}$' which is h(h(ID||x)||SID*||$K_1$+2), from $C_{11}$ by XORing $C_{11}$ and server information h(h(SID||y)||ID*||$K_2$+2). Here, the server increases value of $K_2$ by 1 only as it has been incremented by 1 in $C_{10}$ already. The computation of $C_{15}$ is following,

$$C_{15} = C_{11} \oplus (h(h(SID||y)||ID^*||K_2+2)$$

After the extraction of $C_{15}$, the server randomly chooses b'ϵ $Z^*_p$ and computes two new values, '$C_{16}$' which is a modulus operation 'p' on the value obtained by raising 'g' to the power 'b'' and '$C_{17}$' that is a hashing operation on the XORed value of '$C_{16}$' and the extracted value '$C_{15}$'.

$$C_{16} = (g^{b'})(\text{mod } p)$$

$$C_{17} = h(C_{16} \oplus C_{15})$$

However, the server sends $C_{17}$ to the user through the public network and keeps $C_{16}$ and $C_{15}$ for further computation.

**B.3 $U \rightarrow S_j$: $e_1$**

When user receives $C_{17}$, user XOR the received $C_{17}$ and user information 'h(h(ID||x)||SID*||$K_1$+2)' to compute $C_{16'}$. After extracting $C_{16'}$, user concatenates $C_{16'}$ and $C_{13}$ and performs hash function on the concatenated value. User computes this value as '$e_1$'.

$$C_{16'} = C_{17} \oplus h(h(ID||x)||SID^*||K_1+2)$$

$$e_1 = h(C_{16'}||C_{13})$$

On the computation of $e_1$, user sends the computed value of $e_1$ to target server $S_j$ for mutual authentication and finally session key generation through the public network.

**B.4 $S_j \rightarrow$ U: $e_2$**

Concurrently to the above step 'B.3', the server performs the following step. On receiving $C_{14}$ from user, the server $S_j$ retrieves $C_{13'}$ by XORing $C_{14}$ and server information '$h(h(SID||y)||ID^*||K_2+2)$'.

$$C_{13'} = C_{14} \oplus h(h(SID||y)||ID^*||K_2+2)$$

After computing $C_{13'}$, server computes $e_2$ which is a hashed value calculated by performing hash operation on the concatenated value of $C_{13'}$ and $C_{16}$.

$$e_2 = h(C_{13'}||C_{16})$$

Server also sends '$e_2$' value through public network to user for their mutual authentication and generation of session key between them.

**B.5 Mutual Authentication and Session Key**

This is the final and last step of the scheme. In this step, the User and server authenticates each other and generate a session key between them for their secure communication over the public network using that with any kind of symmetric cryptographic technique. However, the session key generated is only for this session between the user and the target server. For a session key with another server, the user have to login again and go through all the above steps of last two phases for the generation of new session key. The authentication and generation of session key is as follows, At the recipient of $e_1$ and $e_2$ at the respective ends, user and target server concurrently compute $e_3$ and $e_4$ respectively. Here, $e_3$ is a hash value produced by performing hash function on the concatenated value of $C_{13}$ and $C_{16'}$ and $e_4$ is also produced in the same way as $e_3$ but instead of $C_{13}$ and $C_{16'}$, $e_4$ used concatenation of $C_{16}$ and $C_{13'}$, that is $C_{13}$ and $C_{16'}$ in reverse order.

$$e_3 = h(C_{13} || C_{16'})$$

$$e_4 = h(C_{16} || C_{13'})$$

After computing $e_3$ and $e_4$, user and target server evaluate these computed values with the respectively received values, that is $e_2$ and $e_1$.

$$e_3 = e_2?$$

$$e_4 = e_1?$$

If they both come equal, user and target server authenticate the identity of each other and agree to produce a session key between them. They both produce the session key individually at their own ends. However, they still compute same session key.

But if either of them evaluate to different value, user and target server fails each other authenticity and do not proceed for session key generation.

The Session Key defined between User 'U' and Target Server '$S_j$' is,

**Session Key:** $h(h(h(ID||x)||SID||K_1+2)||h(h(SID||y)||ID||K_2+2)||C_{13}+2||C_{16}+2)$

**User** **Authentication Centre** **Target Server**

**A. Authentication Center authenticating User and Server**

Enter ID*, Pw*, SID*

Extract: $R_u=C_0 \oplus h(Pw^* \oplus K)$

Choose: $a \in Z^*_p$

Compute:

$C_1=(g^a)(\text{mod } p)$

$C_2=h(R_u||SID^*||C_1)$

$\xrightarrow{ID^*,\ SID^*,\ C_1, C_2}$

Receive messages from User

$\xrightarrow{ID^*}$

Choose: $b \in Z^*_p$

Compute:

$C_3= (g^b)(\text{mod } p)$

$C_4=h(h(SID||y)||ID^*||C_3)$

$\xleftarrow{ID^*,\ SID^*,\ C_3,\ C_4}$

Receive messages from Server

Verify:

$h(h(ID^*||x)||SID^*||C_1)=C_2?$

$h(h(SID^*||y)||ID^*||C_3)=C_4?$

Choose: $c \in Z^*_p$; $d \in Z^*_p$

$C_5=(g^c)(\text{mod } p)$

$K_1=(C_1)^c=(g^{ac})(\text{mod } p)$

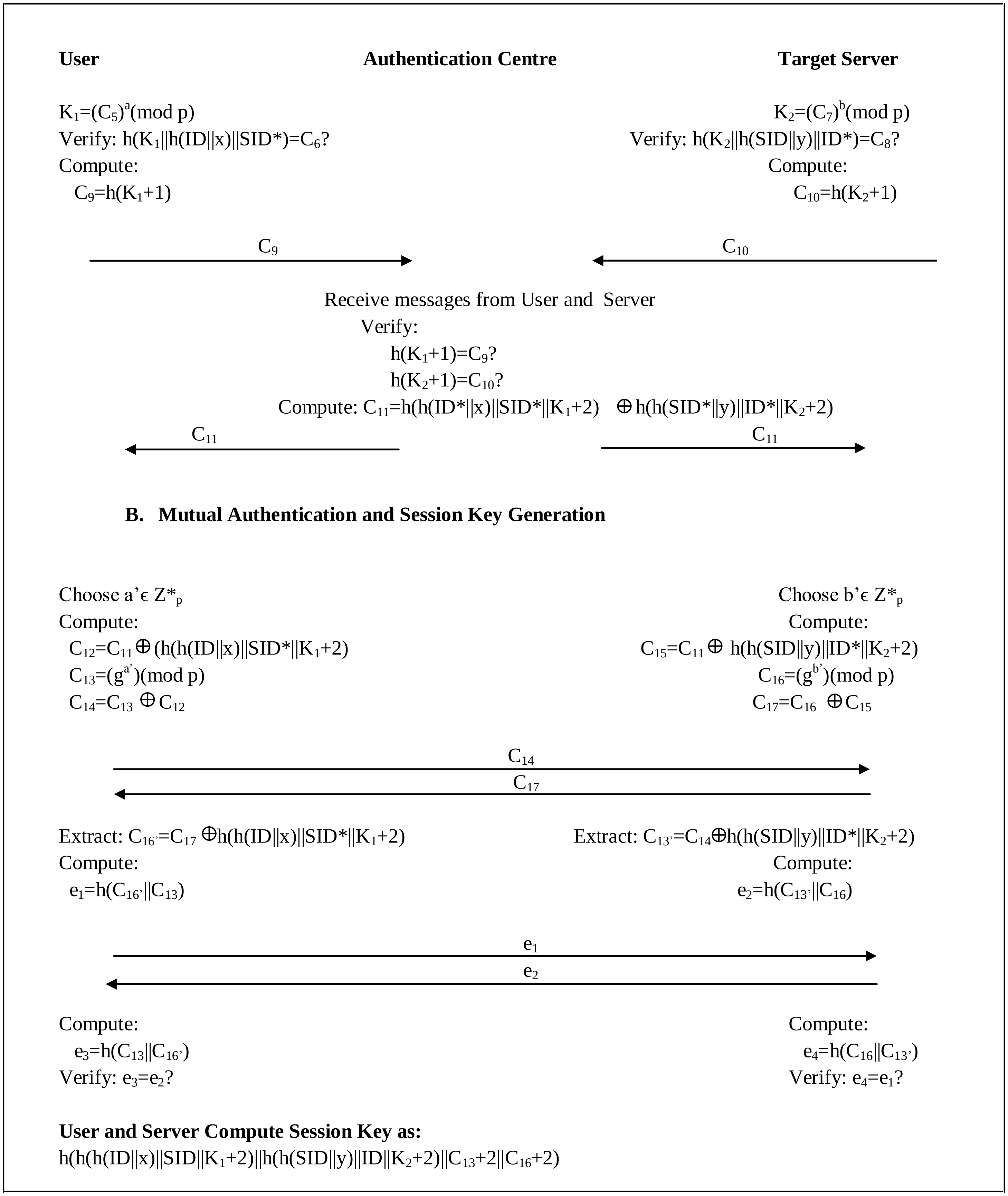
$C_6=h(K_1||h(ID^*||x)||SID^*)$

$C_7=(g^d)(\text{mod } p)$

$K_2=(C_3)^d=(g^{bd})(\text{mod } p)$

$C_8=h(K_2||h(SID^*||y)||ID^*)$

$\xleftarrow{C_5,\ C_6}$ $\xrightarrow{C_7,\ C_8}$



**User** **Authentication Centre** **Target Server**

User: $K_1=(C_5)^a (\text{mod } p)$
Verify: $h(K_1||h(ID||x)||SID^*)=C_6$?
Compute:
$C_9=h(K_1+1)$

Target Server: $K_2=(C_7)^b (\text{mod } p)$
Verify: $h(K_2||h(SID||y)||ID^*)=C_8$?
Compute:
$C_{10}=h(K_2+1)$

User $\xrightarrow{C_9}$ Authentication Centre $\xleftarrow{C_{10}}$ Target Server

Receive messages from User and Server
Verify:
$h(K_1+1)=C_9$?
$h(K_2+1)=C_{10}$?
Compute: $C_{11}=h(h(ID^*||x)||SID^*||K_1+2) \oplus h(h(SID^*||y)||ID^*||K_2+2)$

User $\xleftarrow{C_{11}}$ Authentication Centre $\xrightarrow{C_{11}}$ Target Server

**B. Mutual Authentication and Session Key Generation**

User: Choose $a' \epsilon Z^*_p$
Compute:
$C_{12}=C_{11} \oplus (h(h(ID||x)||SID^*||K_1+2)$
$C_{13}=(g^{a'})(\text{mod } p)$
$C_{14}=C_{13} \oplus C_{12}$

Target Server: Choose $b' \epsilon Z^*_p$
Compute:
$C_{15}=C_{11} \oplus h(h(SID||y)||ID^*||K_2+2)$
$C_{16}=(g^{b'})(\text{mod } p)$
$C_{17}=C_{16} \oplus C_{15}$

User $\xrightarrow{C_{14}}$ Target Server
User $\xleftarrow{C_{17}}$ Target Server

User: Extract: $C_{16'}=C_{17} \oplus h(h(ID||x)||SID^*||K_1+2)$
Compute:
$e_1=h(C_{16'}||C_{13})$

Target Server: Extract: $C_{13'}=C_{14} \oplus h(h(SID||y)||ID^*||K_2+2)$
Compute:
$e_2=h(C_{13'}||C_{16})$

User $\xrightarrow{e_1}$ Target Server
User $\xleftarrow{e_2}$ Target Server

User: Compute:
$e_3=h(C_{13}||C_{16'})$
Verify: $e_3=e_2$?

Target Server: Compute:
$e_4=h(C_{16}||C_{13'})$
Verify: $e_4=e_1$?

**User and Server Compute Session Key as:**
$h(h(h(ID||x)||SID||K_1+2)||h(h(SID||y)||ID||K_2+2)||C_{13}+2||C_{16}+2)$

Figure 3.4: Authentication Phase

# CHAPTER 4: ANALYSIS

In this, we analyze the proposed system in terms of security and performance based on the observations derived from different works in this field. The security analysis validates the methodology against every possible security attack and performance analysis evaluates it in terms of accuracy and efficiency.

## 4.1 Security Analysis

**Theorem 1:** The proposed scheme can withstand against Password Guessing Attack

**Proof:** The proposed scheme does not get attack by on-line password guessing attack as the authentication center will authenticate the user before allowing it for mutual authentication with the server, i.e. after A.5 the password guessing attack will fail. The offline password guessing attack will fail as the password is only protecting the user and no important information is generated by using password. The user with the legal password can only generate the correct value of Ru from the $C_0 \oplus h(Pw^* \oplus K)$ which is contained in the smart card, otherwise the Ru generated by the offline guessed password will be incorrect.

**Theorem 2:** The proposed scheme can withstand against Replay Attack

**Proof:** In the proposed scheme, the replay attack cannot occur at any point of time as the messages transmitted among server, user and authentication center contain the element of freshness in them. The freshness in messages is because of the use of randomly chosen values of a, b, c, d, a’, b’ from $Z^*_p$. So, the proposed protocol can withstand the replay attack till the attacker has not know h(SID||y) or h(ID||x).

**Theorem 3:** The proposed scheme can withstand against Impersonation Attack

**Proof:** The proposed protocol first uses the authentication center to authenticate the server and user and provide them with new authentication key every time they go for mutual authentication phase. If the attacker imitates as the valid user, he cannot get authentication from authentication center without knowing user h(ID||x), $C_1$ and $C_2$. The server is provided only the user id for computation. Thus, the attacker will not get the correct authentication key. Thus, the proposed protocol resists Impersonation attack.

**Theorem 4:** The proposed scheme can withstand against Insider Attack

**Proof:** In the proposed scheme, the user makes registration to authentication center by presenting its password in the form ($Pw \oplus K$) instead of simply providing it as (Pw). So, any insider in the authentication center won’t be able to get to know the actual user password till it does not know the value of K, which is a randomly generated value. So, the proposed scheme successfully resists insider attack.

**Theorem 5:** The proposed scheme can withstand against Stolen-Verifier Attack

**Proof:** The proposed scheme does not allow authentication center and server to hold any verification table. Due to this, the stolen-verifier attack is impossible to occur. The authentication center and server authenticates the user from the values it provides to them.

**Theorem 6:** The proposed scheme can withstand against Man-In-The-Middle Attack

**Proof:** Since the message generated by the server and user for the authentication center contains the secret identity of the user and server, the message generated by the adversary would fail to get authentication by the authentication center. The adversary would not know the values of ‘x’ and ‘y’. So, the messages generated by the adversary cannot imitate the correct values of user and server. Thus, our scheme resists Man-in-the-Middle attack.

**Theorem 7:** The proposed scheme can withstand against Server Spoofing Attack

**Proof:** In the proposed protocol, the attacker cannot masquerade as $S_j$ and cheat user or authentication center. As servers do not contain verification table, they cannot authenticate any user directly. To authenticate the user, the server must get authenticated by the authentication center first and then have the $h(ID||x)$ of user. The attacker must know $h(SID||y)$ to cheats authentication center. Therefore, this scheme resists Server Spoofing attack.

**Theorem 8:** The proposed scheme can withstand against Authentication Center Spoofing Attack

**Proof:** In our scheme, the attacker cannot masquerade as authentication center, as every user and server has $(ID||x)$ and $(SID||y)$ respectively. So, they use their secret information in $C_6$ and $C_8$ to authenticate the authentication center. Thus, this scheme resists Authentication Center Spoofing Attack.

**Theorem 9:** The proposed scheme can withstand against Forward Secrecy Attack

**Proof:** The scheme is forward secure. Even if x and y discloses, the attacker has to get two messages $C_1$, $C_2$, $C_3$ and $C_4$ to get authenticate from the authentication center. Since, $C_1$ and $C_3$ are randomly chosen. The attacker will not able to guess these accurately and thus generate access to session key. So, the attacker, getting only secret key x and y, cannot get the session key.

**Theorem 10:** the proposed scheme can withstand against Denning-Sacco Attack

**Proof :** The Denning-Sacco attack is where an attacker compromises an old session key and tries to find a long-term private key (e.g. user password or server private key) or other session keys[12]. The proposed scheme resists this type of attack as it is harder for the attacker to get the $K_1$ and $K_2$ and spoof other entities which are in communication with it.

**Theorem 11:** The proposed scheme provides Authentication Key Security

**Proof:** in this scheme, the authentication key cannot be calculated by anyone other than the server, user and authentication center since it contains random values $K_1$, $K_2$, user secret key and server secret key. The attacker can only get authentication key if it gets to know these values which are random and very large.

**Theorem 12:** The proposed protocol provides Security of Session Key

**Proof:** In this scheme, the session key cannot be calculated by any other except the server and user since it contains random values $C_{13}$, $C_{16}$, user authentication key, server authentication key. Only, the authenticated server and authenticated user can generate these values and the session key. Thus, the session key is secure.

| **Security Properties** | **Proposed Scheme** |
|---|---|
| Password Guessing Attack | Yes |
| Replay Attack | Yes |
| Impersonation Attack | Yes |
| Insider Attack | Yes |
| Stolen-Verifier Attack | Yes |
| Man-In-The-Middle Attack | Yes |
| Server Spoofing Attack | Yes |
| Authentication Center spoofing Attack | Yes |
| Forward Secrecy Attack | Yes |
| Denning-Sacco Attack | Yes |
| Authentication Key Security | Yes |
| Session Key Security | Yes |

Table 4.1: Security Properties

## 4.2 Performance Analysis

We have implemented the authentication scheme on Intel(R) Core™ i3-2330 CPU @ 2.20 GHz and window 7 operating system using C# and Visual Studio2010.

We have successfully implemented this scheme on the above mentioned platform and the results are convincing. It has generated the session key between a remote user and a server in the distributed environment securely. It does not authenticate invalid user and invalid server before computing session key where the invalid criteria include wrong user password, wrong user id, wrong server id. Our implementation rejects any blank inputs and puts off the server with which the user is not asking for session key.

Since the scheme uses Diffie-hellman key Exchange agreement, the generator of $Z_p*$ is usually taking a considerable time in computation when the number of digit of p exceeds five. The authentication scheme must take computation and efficiency into consideration. In our scheme, user needs three XOR, ten concatenations and seven hash function operations; also, two XOR, ten concatenation and six hash function operations are performed by the server. The authentication performs one XOR, eighteen concatenation and ten hash function operations. As only the hash function is taken in computation cost, our scheme computation is considerable. But, hash-function operation is done with large computation system and smart card is used to store the values, thereby the computation cost is not a constraint. As compare to [11], the scheme is more efficient and computationally less costly.

This Chapter presented analysis on the methodology proposed in this thesis. This explains in detail how it resists various security attacks and efficient.

# CHAPTER 5: IMPLEMENTATION OF PROPOSED METHODOLOGY

This chapter includes the implementation of the proposed methodology, which describes how the methodology can be practically implemented. The implementation requires a computer with network capabilities. This computer can be used to demonstrate the program alone, if it can handle two simultaneous servers, authentication center and a client working at the same time. Additionally to demonstrate the full strength, two or more computers will be required, that can work as servers with full network capabilities. All ends were coded in C#. So, .Net framework is required on the machines to run the program. No additional hardware requirement is there.

Kindly note, the implementation is from a single developer’s point of view. Different developers may use different approaches to implement the given model. It all depends upon the developers comfort level with a particular set of tools/languages.

## 5.1 Development Environment

In this, we introduced the various technologies that were used while designing a practical working model for the demonstration of the concept. The description includes only necessary details of the required technology in the project. Most of the technologies used in the project are freely available on the internet.

### 5.1.1 .Net Framework

The .NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for the .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

The .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces a popular integrated development environment largely for .NET software called Visual Studio [43].

### 5.1.2 Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio supports different programming languages that includes C/C++, VB.NET, C#, and F#. Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It also supports XML/XSL, HTML/XHTML, JavaScript and CSS [44].

### 5.1.3 C#

C# is a multi-paradigm programming language encompassing strong imperative, declarative, functional, generic, object-oriented and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure [45].

C# is intended to be a simple, modern, general-purpose, object-oriented programming language.[6] Its development team is led by Anders Hejlsberg. The design goals for C# are:

- C# language is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Source code portability is very important, as is programmer portability, especially for those programmers already familiar with C and C++.

## 5.2 Procedures in Implementation

To show the working of the methodology, we have stimulated smart card and the smart card terminal as a windows application and the target server also, with which the user wants to communicate. The authentication center has been stimulated as a console application. The procedure for various modules used in the implementation is as follows. The code for the procedures can be found in the Appendix-I.

### 5.2.1 Procedure for Group Generator

This procedure is used for generating the generator of a cyclic group. This is required for the selection of parameters and performing modular operations.

```
primit_root(Prime_no, fhi)

Here, prime_no is ramdomly chosen prime number and fhi is a Euler's Totient function
that gives the order of the group.

1 Intialise re, i, a.
2 Repeat steps  3 to 6 while a < (Prime_no - 1)
3         Repeat steps 4 and5 while re !=1
4                 Assign re =  pow(a,i) mod Prime_no.
5                 Increament i till re != 1.
6 If i  is equal to fhi ,
7         break
8 return a.
```

### 5.2.2 Procedure for Hash and XORing Operation

The following procedure performs a hash operation on the concatenated string of numbers and words. However, it generates output in an integer number which is further XORed with another integer used in further calculations.

```
1 Concatenate string and number to a string.
2 Extract Ascii code of the concatenated string
3 Perform hashing on the ascii code.
4 Convert Ascii code to integer .
5 Perform xor on the two integer numbers.
```

### 5.2.3 Procedure for Power Operation using Square and Multiply Method

This procedure is used for performing the power operation on binary form of the number. It uses square and multiply method for the easy and fast computation.

```
Pow(int e, int f, int n)

1 Initialize x
2 Assign binary form of f to x
3 Set y = sq_and_multi(a, x, n).
4 Return y.

//---------- the square and multiply method --------------//

sq_and_multi(a, x, n)

1 y = 1
2 //as nb is the number of bits in x.
3 For (i=0 to nb-1 )
4 {
5           // multiply only if the bit is only 1.
6           If ($x_i$ = 1)
7           y = (a*y) mod n
8           // squaring is not needed in the last iteration.
9           a = $a^2$ mod n
10  }
11 return y

//------ to convert integer to binary ------------//

1 // N is the decimal number
2 While N > 0
3 // output the remainder
4 Print N mod 2
5 // replace N by n divided by 2
6 N = N/2
```

## 5.2.4 Procedure for Sending Data

This procedure creates a socket and then sends the string over the public network. We have used a fixed port for simplicity.

```
// Here, port is used for port number and sck_r is for the socket at the client side.
port = 7798.
// Define Socket
sck_r = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp).
IPEndPoint localendpoint = new IPEndPoint(IPAddress.Parse(aip_r), port).
Initialise data to text.
try
{
    sck_r.Connect(localendpoint);
    cx = "User is connected to authencation centre for registration ";
    Send data.
}
```

### 5.2.5 Procedure for Receiving Data

This procedure creates a socket and then accepts the data over the public network. We have used a fixed port for simplicity.

```
// Here, port is used for port number and sck2 is for the socket at the authentication side.
port = 1235;
// Define Socket.
sck2 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
sck2.Bind(new IPEndPoint(0, pno));
sck2.Listen(1);
// Receive Data
Accept the socket request.
Receive the data through socket.
```

### 5.2.6 Procedure for Mutual Key

This procedure transmits the mutual key over the public network which is used by the server and user to generate session key. Before sending the key, it authenticates the validity of user and server and security of the public channel.

```
Generate a by concatenating user id, server id and user token.
Perform hash operation on a.
Store the hashed value in a1.
Generate b by concatenating server id, user id and server token.
Perform hash operation on b.
Store the hash value in b1.
Perform xor-ing in a1 and b1.
```

### 5.2.7 Procedure for Session Key

The procedure below generates the session key at the user and server end individually. However, the generated key is same. This key is used for communication over the public network.

```
1 Perform hashing on concatenated value of user id, server id and user token.
2 Store the hashed value in a.
3 Perform hashing on concatenated value of server id, user id and server token.
4 Store the hash value in b.
5 Increment c and d, diffie-hellman intermediates by 2.
6 Concatenate a,b,c,d.
7 Perform hash function on concatenated value .
```

### 5.2.8 Procedure for Encrypting Data

This procedure is used to encrypt the data to send to server over the public network. The key for encryption process is the session key generated between user and server.

Encryption: $C_i=(P_i + K) \bmod 26$

where, Ci is the ciphertext, Pi is the plaintext, K is user session key

### 5.2.9 Procedure for Decrypting Data

This procedure is used to decrypt the encrypted data received from user over the public network. The key for decryption is the session key generated between user and server.

Decryption: $P_i=(C_i-K+26) \bmod 26$

where, Ci is the ciphertext, Pi is the plaintext, K is server session key

### 5.2.10 Procedure for Diffie-Hellman Key Exchange

This is a sample procedure for Diffie Hellman Key Exchange Scheme.

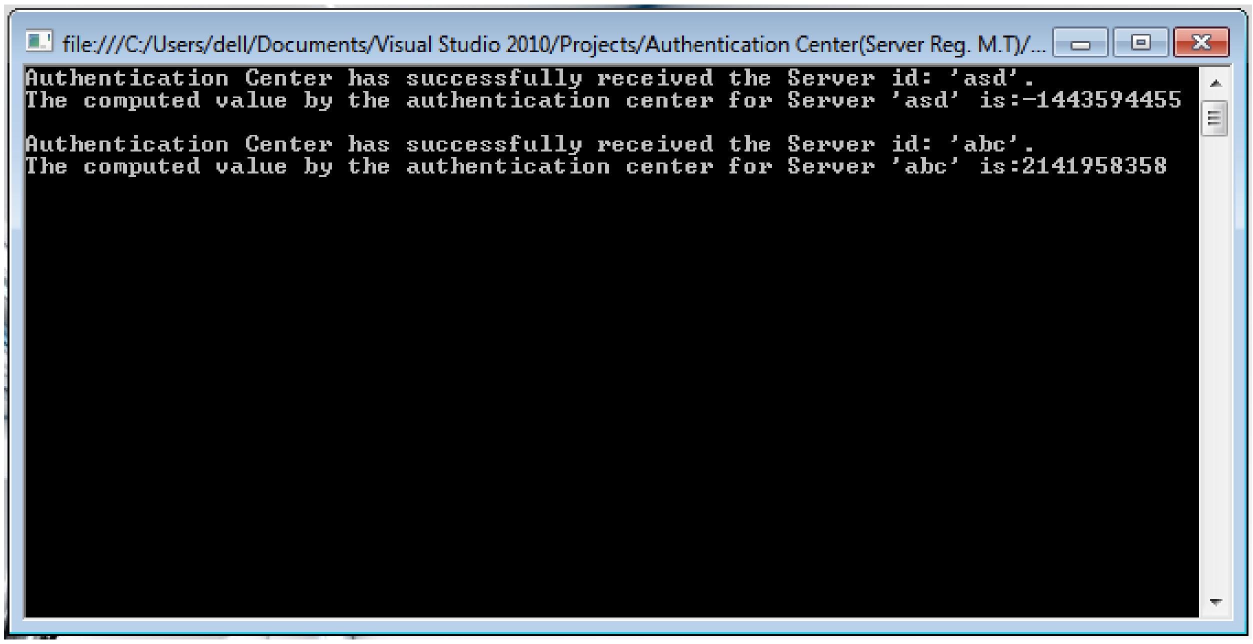
```
1 // At user end
2 Choose randomly x.   //0<x<p-1
3 Calculate $c_1 = g^x \bmod p$.
4 //At server end
5 Choose randomly y.  //0<y<p-1
6 Calculate $c_2 = g^y \bmod p$.
7 User sends $c_1$ to server.
8 Server sends $c_2$ to user.
9 User calculates $K = (c_2)^x \bmod p$.
10Server calculates $K= (c_1)^y \bmod p$.
```

## 5.3 Results

This section gives us the implementation results of the project in form of screenshots. The Screen-shots have brief description about them. This helps us to visualize the project better. This also gives us a fair idea of how the system will work, if implemented on a larger scale. The results were calculated on a single machine which was capable of running the two servers, authentication center and the client.

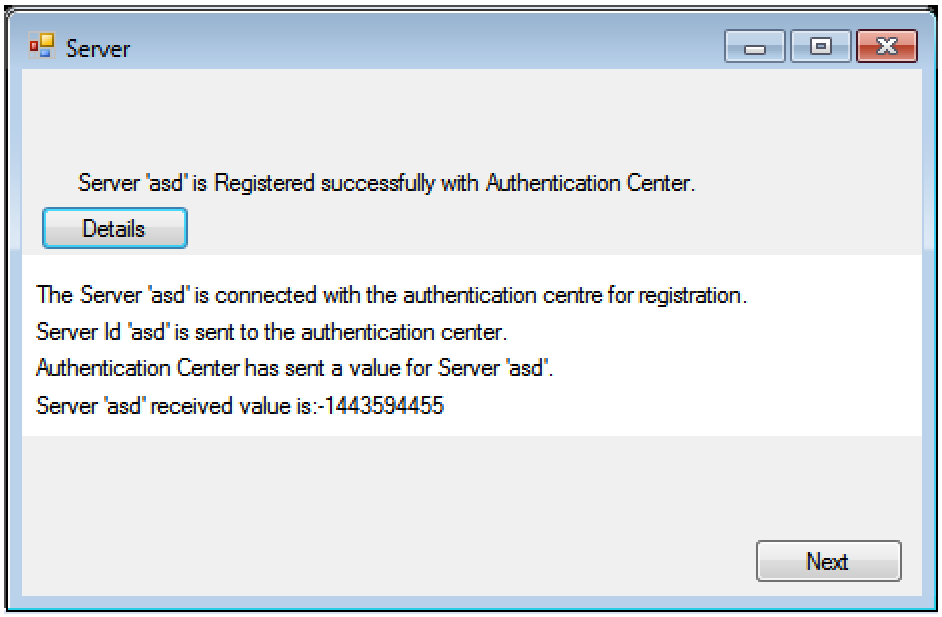
### 5.3.1 Server Registration:

In this, the two servers register themselves with the authentication center by connecting and sending their IDs' to the authentication center. On receiving the servers IDs, the authentication center performs the computation of phase 1 of the methodology. The results of the computation are shown below, when two servers with ID 'asd' and 'abc' are getting register themselves to the authentication center.


file:///C:/Users/dell/Documents/Visual Studio 2010/Projects/Authentication Center(Server Reg. M.T)/...

```
Authentication Center has successfully received the Server id: 'asd'.
The computed value by the authentication center for Server 'asd' is:-1443594455

Authentication Center has successfully received the Server id: 'abc'.
The computed value by the authentication center for Server 'abc' is:2141958358
```

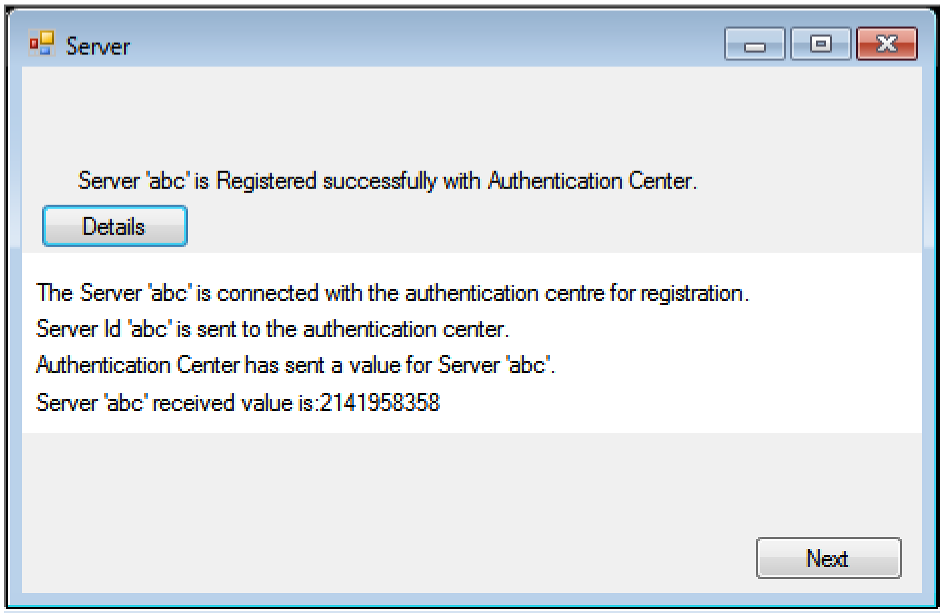
Figure 5.1: Authentication Center(Server Registration)



Server 'asd' is Registered successfully with Authentication Center.

The Server 'asd' is connected with the authentication centre for registration.
Server Id 'asd' is sent to the authentication center.
Authentication Center has sent a value for Server 'asd'.
Server 'asd' received value is:-1443594455

Figure 5.2: Server ‘asd’ Registration



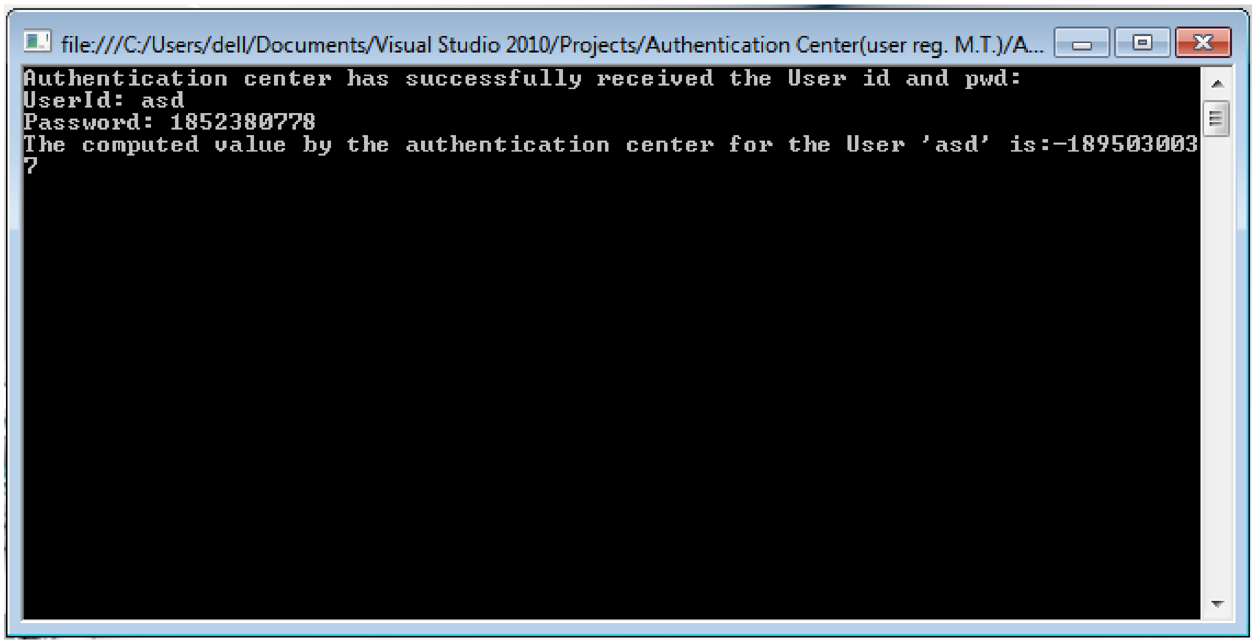
Server 'abc' is Registered successfully with Authentication Center.

```
The Server 'abc' is connected with the authentication centre for registration.
Server Id 'abc' is sent to the authentication center.
Authentication Center has sent a value for Server 'abc'.
Server 'abc' received value is:2141958358
```

Figure 5.3: Server 'abc' Registration

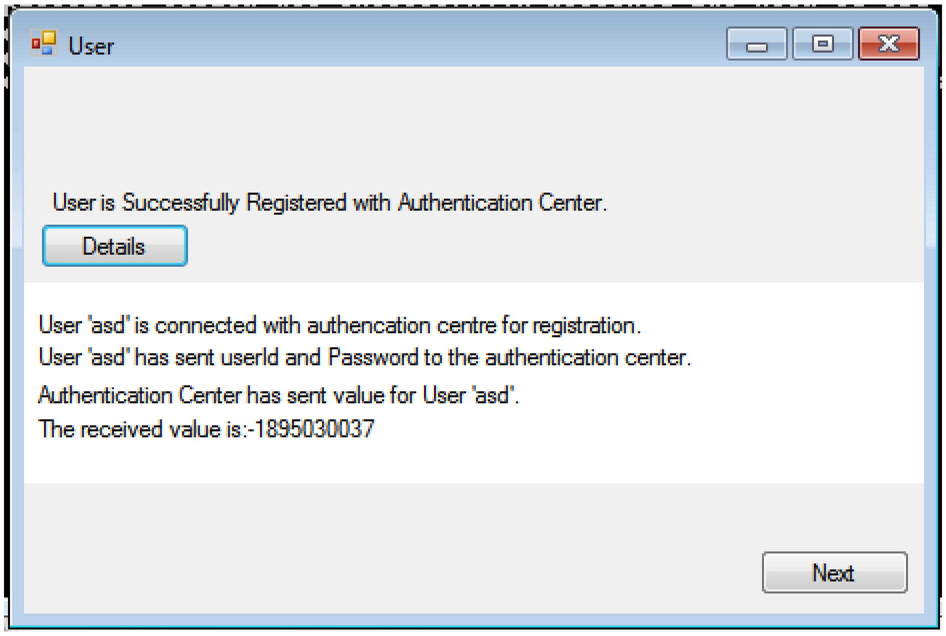
### 5.3.2 User Registration:

During this phase, the user registers at the authentication center by sending user ID and password. The authentication center performs the operations of phase 2 on the received data from the user. Here, there are snapshots of the results when user 'asd' with password 'asd' registers at the authentication center successfully.


file:///C:/Users/dell/Documents/Visual Studio 2010/Projects/Authentication Center(user reg. M.T.)/A...

```
Authentication center has successfully received the User id and pwd:
UserId: asd
Password: 1852380778
The computed value by the authentication center for the User 'asd' is:-189503003
7
```

Figure 5.4: Authentication Center(User Registration)



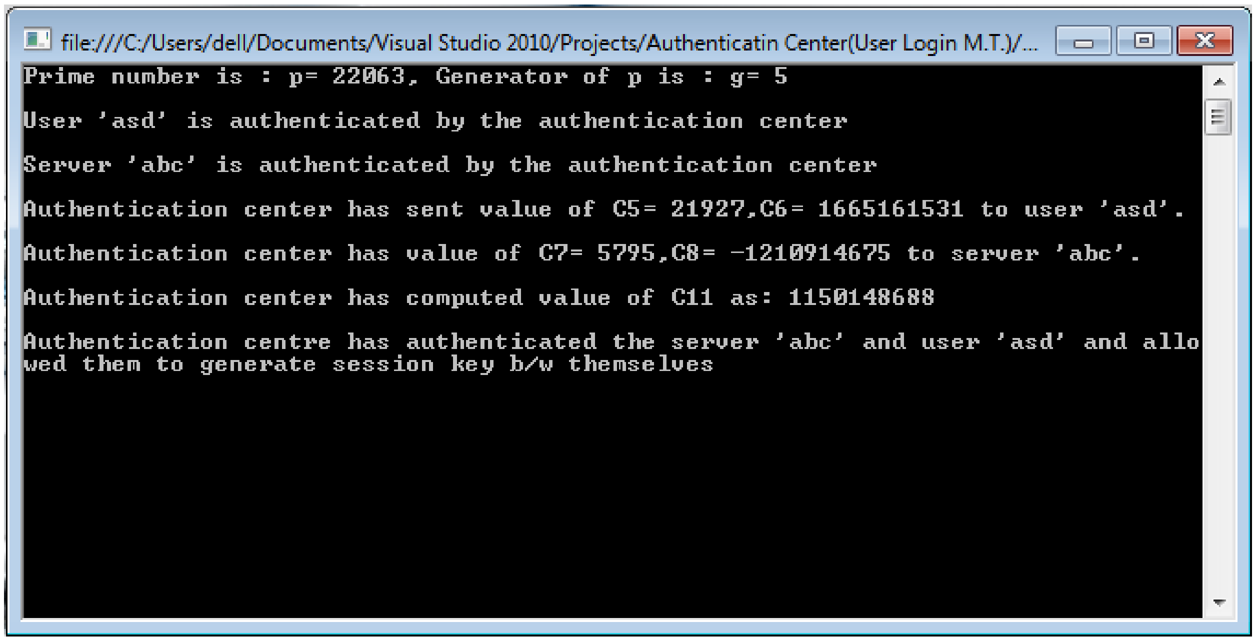
User is Successfully Registered with Authentication Center.

User 'asd' is connected with authencation centre for registration.
User 'asd' has sent userId and Password to the authentication center.
Authentication Center has sent value for User 'asd'.
The received value is:-1895030037

Figure 5.5: User Registration

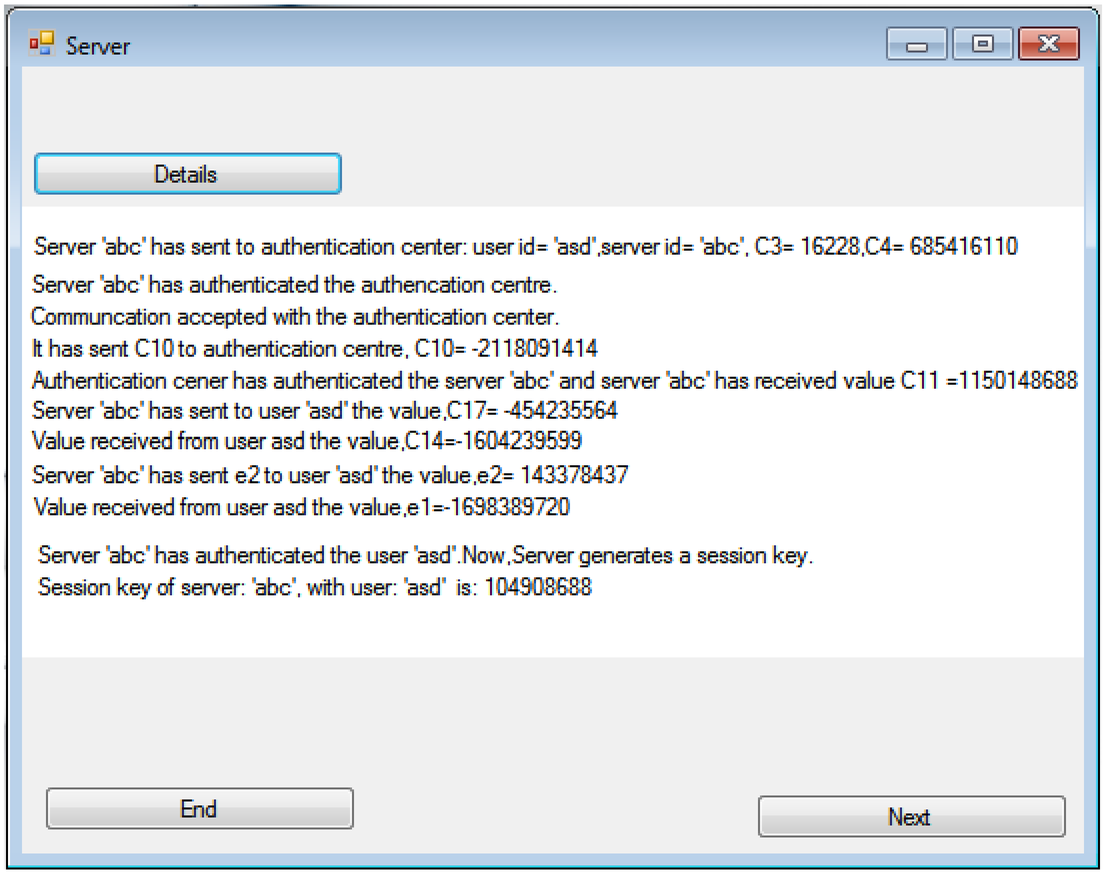
### 5.3.3 Authentication of Remote User and Server and Mutual Authentication and Session Key Generation:

During this phase, the user logins at the authentication center with its unique ID and password, received during the user registration phase. Before the user login, all the servers registered with the authentication center login and make themselves available to the use of user. After user successfully logins, the user enters the ID of the target server with which user wants to communicate. Then, the authentication center authenticates the user and the target server using the operation in phase 3. In this phase, the user and the target server are also meant to authenticate the authentication center. After authentication of each other, the AC allows user and the target server to authenticate each other directly and communicate for the generation of session key between each other using the operations of phase 4 of our scheme. The user ‘asd’ has chosen server ‘abc’ for communication. Here are the snapshots of the communication among the AC, target server ‘abc’ and user ‘asd’ for the authentication and generation of session key.


file:///C:/Users/dell/Documents/Visual Studio 2010/Projects/Authenticatin Center(User Login M.T.)/...

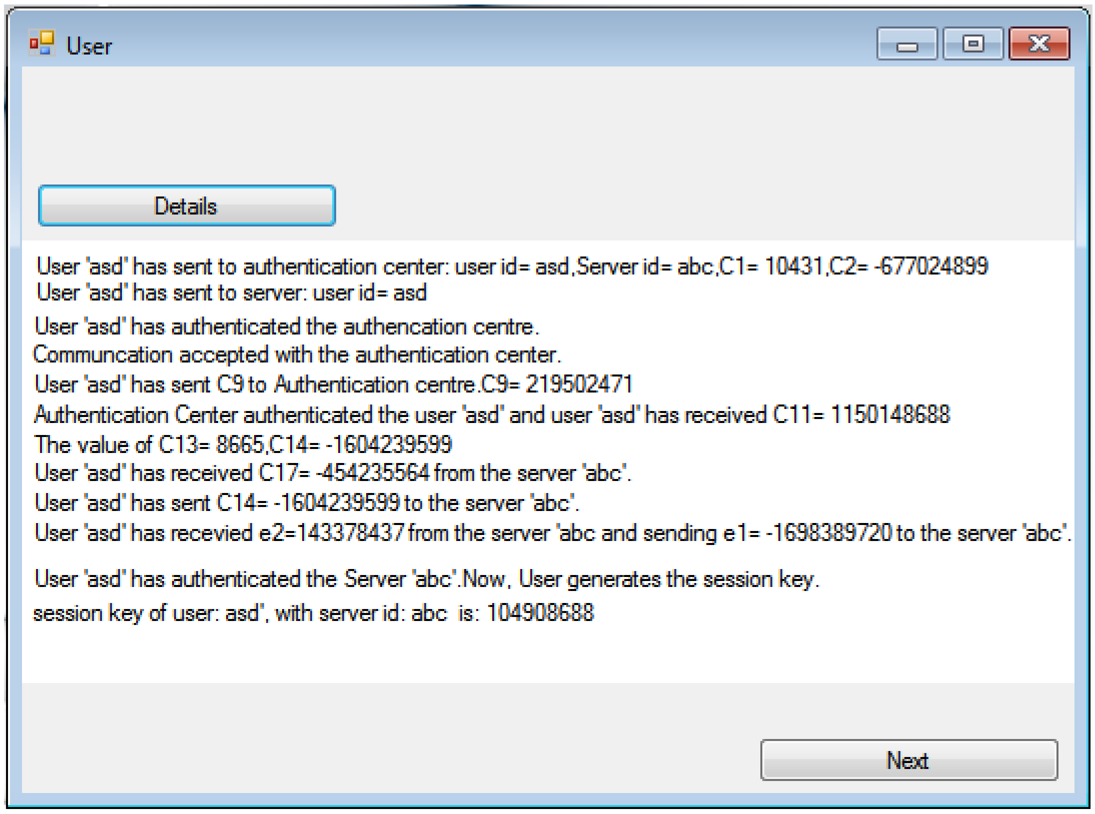
```
Prime number is : p= 22063, Generator of p is : g= 5

User 'asd' is authenticated by the authentication center

Server 'abc' is authenticated by the authentication center

Authentication center has sent value of C5= 21927,C6= 1665161531 to user 'asd'.

Authentication center has value of C7= 5795,C8= -1210914675 to server 'abc'.

Authentication center has computed value of C11 as: 1150148688

Authentication centre has authenticated the server 'abc' and user 'asd' and allo
wed them to generate session key b/w themselves
```

Figure 5.6: Authentication Center



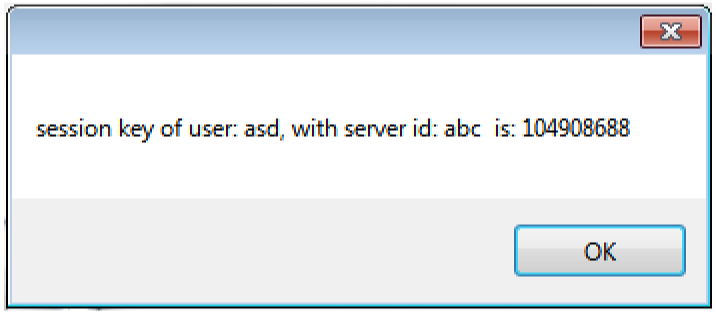
```
Server 'abc' has sent to authentication center: user id= 'asd',server id= 'abc', C3= 16228,C4= 685416110
Server 'abc' has authenticated the authencation centre.
Communcation accepted with the authentication center.
It has sent C10 to authentication centre, C10= -2118091414
Authentication cener has authenticated the server 'abc' and server 'abc' has received value C11 =1150148688
Server 'abc' has sent to user 'asd' the value,C17= -454235564
Value received from user asd the value,C14=-1604239599
Server 'abc' has sent e2 to user 'asd' the value,e2= 143378437
Value received from user asd the value,e1=-1698389720

Server 'abc' has authenticated the user 'asd'.Now,Server generates a session key.
Session key of server: 'abc', with user: 'asd'  is: 104908688
```

Figure 5.7: Target Server Session Key



```
User 'asd' has sent to authentication center: user id= asd,Server id= abc,C1= 10431,C2= -677024899
User 'asd' has sent to server: user id= asd
User 'asd' has authenticated the authencation centre.
Communcation accepted with the authentication center.
User 'asd' has sent C9 to Authentication centre.C9= 219502471
Authentication Center authenticated the user 'asd' and user 'asd' has received C11= 1150148688
The value of C13= 8665,C14= -1604239599
User 'asd' has received C17= -454235564 from the server 'abc'.
User 'asd' has sent C14= -1604239599 to the server 'abc'.
User 'asd' has recevied e2=143378437 from the server 'abc and sending e1= -1698389720 to the server 'abc'.
User 'asd' has authenticated the Server 'abc'.Now, User generates the session key.
session key of user: asd', with server id: abc  is: 104908688
```

Figure 5.8: User Session Key

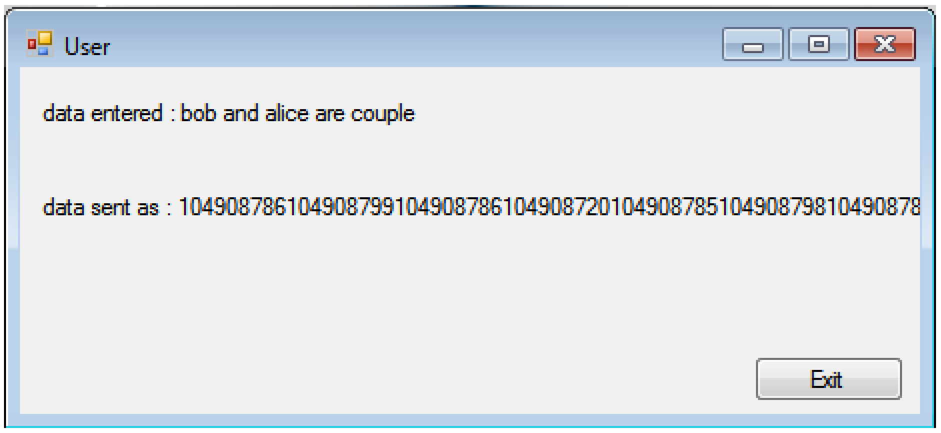


session key of user: asd, with server id: abc  is: 104908688

Figure 5.9: Session Key Generated.

The value of public variables ‘p’ and ‘g’ are 26713 and 10 respectively. The values of various variables in communication among authentication center, target server ‘abc’ and user ‘asd’ shown in the above snapshots. However, the session key generated between target server ‘abc’ and user ‘asd’ is 104908688.

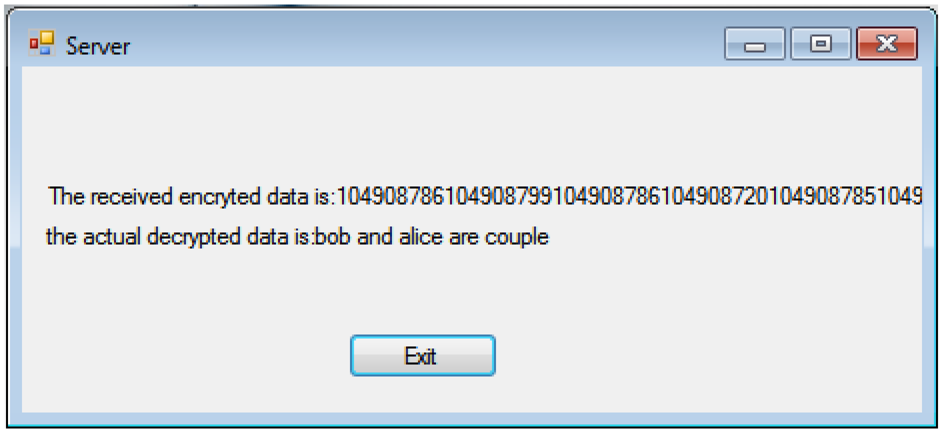
The user ‘asd’ has used this session key to encrypt the data ‘bob and alice are couple’ and sent it to the server ‘abc’. This data transmission is successfully done as seen in the figure 5.10 and 5.11 below. We have used simple cesar cipher technique to encrypt and decrypt this data transmission.



data entered : bob and alice are couple

data sent as : 1049087861049087991049087861049087201049087851049087981049087

Figure 5.10: User encrypting data using session key



The received encryted data is:104908786104908799104908786104908720104908785104908

the actual decrypted data is:bob and alice are couple

Figure 5.11: Server decrypting data using session key

This chapter has shown the implementation and outputs of the implemented project. The Code presented in this discussion is not exactly the same code used in my project but a similar code that is easier to understand. We have also discussed the specifications for the proposed methodology which gives us the fair idea about the machines required for the demonstration of the system. The User smart card and smart card terminal is stimulated as a single window application. The Authentication center's console messages and target server's window messages are also discussed in brief, which helps us to see the actually working at authentication center and the server which is normally hidden. The Chapter has in short, shown the complete internal working of the project which can be analyzed further if required.

# CHAPTER 6: CONCLUSION AND FUTURE WORK

This chapter concludes the work done on the proposed methodology and the future work that can be done to improve our proposal. This chapter is a discussion which is limited to my thinking and analytical power. Different people have different approach to a particular problem. So, some people may not agree to the ideas presented by me in this discussion.

## 6.1 Conclusion of the Thesis

In this, we proposed a new efficient and secure multi-server authentication using one-way hash function, XOR function and Diffie-Hellman. The scheme proposed does not maintain any verification table, allow freely chosen password, low computation and communication cost, mutual authentication, session key agreement, access control and provide security against every possible attack. It does not authenticate invalid user or invalid server before computing the session key. The weakness found in the Zhu et al[123] and Xie and Chen[128] schemes have been successfully resisted by our scheme. The scheme has not used any encryption/decryption operations to establish authentication and session key between the remote user and the server. The analysis of the scheme has proven that it can securely authenticate and generate a session key between the remote user and the requested server in a multi-server distributed environment for communication process.

## 6.2 Future Work

Since the methodology is one mainly designed for smart-card, we can even allow mobile phones, smartcards, PDA, etc. to act as a client. However, the system doesn’t allow these low memory devices. Currently this is not included. The future work also includes more enhancing the efficiency of the methodology. In addition, we can also look for the proposal of a multi-server multi-authentication center authentication schema.

# CHAPTER 7: REFERENCES AND BIBLIOGRAPHY

[1] **Lamport L.**, “Password Authentication with insecure communication”. Communication of the ACM November24(11)(1981)770-772

[2] **Hwang T, Chnen Y, Laih CS**, “Non–interactive password authentication without password tables”, In : IEEE region 10 conference on computer and communication system(1990)429-431

[3] **Wang XY, Yu HG**, “How to break MD5 andother hash functions”, Eurocrypt (2000)19-35

[4] **Sun HM**, “An efficient remote use authentication scheme using smart cards”, IEEE Transactions on Consumer Electronics 46(4)(2000 )958-961.

[5 ] **Li LH, Lin IC, Hwang MS**, “A remote password authentication scheme for multi –server architecture using neural network”, IEEE Transactions on Neural Network 12(06)(2001)1498-1504.

[6] **Eun-Jun Yoon, Kee-Young Yoo***, “Robust Multi-Server Authentication Scheme”, IEEE sixth IFIP International Conference on Network and Parallel Computing(2009).

[7] **Juang WS**, “Efficient multi-server password authenticated key agreement using smart cards”, IEEE Transactions on Consumer Electronics 50(1)(2004)251-255.

[8] **Y. P. Liao, S. S. Wang**, “A secure dynamic ID based remote user authentication scheme for multi-server environment”, Computer Standards & Interfaces, 31(1): 24-29, 2009.

[9] **H. C. Hsiang, W. K. Shih**, "Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment", Computer Standards & Interfaces.

[10] **J. L. Tsai**, "Efficient multi-server authentication scheme based on one-way hash function without verification table", Computers & Security, 27(2008):115-121, 2008.

[11] **Qi Xie and Deren Chen**, "Hash function and smart card based multi-server authentication protocol", IEEE WASE International Conference on Information Engineering,2010.

[12] **H. Zhu, T. Liu, J. Liu**, "Robust and Simple multi-server authentication protocol without verification table", Ninth International Conference on Hybrid Intelligent Systems, 2009.

[13] **TY Chen, MS Hwang*, CC Lee, JK Jan**, "Cryptanalysis of a secure dynamic ID based remote user authentication scheme for multi-server environment", Fourth International Conference on Innovative Computing, Information and Control, 2009.

[14**] MH Shao, YC Chin**, "A Novel Dynamic ID-based Remote User Authentication and Access Control Scheme for Multi-Server Environment", 10th IEEE International Conference on Computer and information Technology(CIT 2010), 2010.

[15] **M.S. Hwang and I.H. Li**, "A new remote user authentication scheme using smart cards", IEEE Transactions on Consumer Electronics, vol. 46, no. I, pp. 28-30, 2000.

[16] **M.L. Das, A. Saxena, and V.P. Gulati**, "A dynamic ID-based remote user authentication scheme", IEEE Transactions on Consumer Electronics, vol. 50, pp. 629-631, 2004.

[17**] H.Y. Chien and C.H. Chen**, "A remote authentication scheme preserving user anonymity", International Conference on Advanced Information Networking and Applications, vol. 2, pp. 245-248, 2005.

[18] **S. Kim, H.S. Rhee, J.Y. Chun and D.H. Lee**, “Anonymous and traceable authentication scheme using smart cards”, International Conference on Information Security and Assurance, pp. 162-165, 2008.

[19] **C.C. Chang and J.Y. Kuo.** “An efficient multi-server password authenticated key agreement scheme using smart cards with access control”, In 19th IEEE int. Conf. Advanced Information Networking and Applications(AINA2005), volume 2, pages 257-260, Taipei, Taiwan, March 2005. IEEE Computer Society.

[20] **R. J. Hwang and S. H. Shiau**, “Provably efficient authenticated key agreement protocol for multi-servers”, The Computer Journal, 50(5):602–615, 2007.

[21] **I. C. Lin,** “A neural network system for authenticating remote users in multi-server architecture”, International Journal of Communication Systems, 21:435–445, 2008.

[22] **I. C. Lin, M. S. Hwang**, and L. H. Li, “A new remote user authentication scheme for multi-server architecture”, Future Generation Computer System January, 19:13–22, 2003.

[23] **W. J. Tsaur, C. C. Wu, and W. B. Lee**, “A smart card based remote scheme for password authentication in multi-server internet services”, Computer Standards and Interfaces, 2004.

[24] **Y. M. Tseng, T. Y. Wu, and J. D. Wu**, “A pairing-based user authentication scheme for wireless clients with smart cards”, INFORMATICA, 19(2):285–302, 2008.

[25] **W. C. Ku**, “Weaknesses and drawbacks of a password authentication scheme using neural networks for multi-server architecture”, IEEE Transactions on Neural Networks, 2005.

[26] **X. Cao, S. Zhong**, “Breaking a remote user authentication scheme for multi-server architecture”, IEEE Communications Letters, Vol. 10, No. 8, pp. 580-581, August 2006.

[27] **W.J. Tsaur**, C.C. Wu, W.B. Lee, “An enhanced user authentication scheme for multi-server Internet services”, Applied Mathematics and Computation, Vol. 170, 258-266, November 2005.

[28] **Z.F. Cao, D.Z. Sun**, “Cryptanalysis and Improvement of User Authentication Scheme using Smart Cards for Multi-Server Environments”, Proceedings of International Conference on Machine Learning and Cybernetics, pp. 2818-2822, August 2006.

[29] **L. Hu, X. Niu, Y. Yang**, “An Efficient Multi-server Password Authenticated Key Agreement Scheme Using Smart Cards”, Proceedings of International Conference on Multimedia and Ubiquitous Engineering, pp. 903-907, April 2007.

[30] **J. H. Lee, D. H. Lee**, “Efficient and Secure Remote Authenticated Key Agreement Scheme for Multi-server Using Mobile Equipment”, Proceedings of International Conference on Consumer Electronics, pp. 1-2, January 2008.

[31] **W. C. Ku, H. M. Chuang, and M. H. Chiang**, “ Cryptanalysis of a multi-server password authenticated key agreement scheme using smart cards”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E88- A(11):3235–3238, November 2005.

[32] **R.C. Wang, W.S. Juang, C.L. Lei**, “User Authentication Scheme with Privacy-Preservation for Multi-Server Environment”, IEEE Communication Letters, no.2, February 2009.

[33] **W.S. Juang, H.C. Tseng and Y.Y. Shue**, “An Efficient and Privacy Protection Multi-server Authentication Scheme for Low-cost RFID Tags”, IEEE 2010.

[34] **Y. L. Chen, C. H. Huang, J. S. Chou**, “A novel multi-server authentication protocol ”, http://eprint.iacr.org/2009/176, 2009.

[35] **B. Shneier**, “Applied Cryptography Second Edition, “John Wiley & Sons, Inc.,1996.

[36] **A.J. Menezes, P.C. Oarschot, and S.A. Vanstone,** "Handbook of Applied Cryptograph," CRC Press, New York, 1997.

[37] **W. Mao**, "Modern  Cryptography Theory & Practice," Prentice Hall, 2004.

[38] **D. Stinson**, "Cryptography Theory and Practice Second edition," Chapman & Hall/CRC, 2002.

[39] **C. Boyd and A. Mathuria**, "Protocols for Authentication and Key Establishment," Springer-Verlag Berlin Heidelberg New York, 2003.

[40] **W. Juang**, C. Lei and C. Chang, "Anonymous Channel and Authentication in Wireless Communications," Computer Communications, Vol. 22,  o. 15-16, pp. 1502-1511, 1999.

[41] **W. Juang**, "Efficient Password Authenticated Key Agreement Using Smart Cards," Computers & Security, in press, 2004.

[42] http://en.wikipedia.org/wiki/Man-in-the-middle_attack, Wikipedia.

[43] http://en.wikipedia.org/wiki/.NET_Framework, Wikipedia.

[44] http://en.wikipedia.org/wiki/Microsoft_Visual_Studio, Wikipedia.

[45] http://en.wikipedia.org/wiki/C_Sharp_(programming_language), Wikipedia.

# APPENDIX A

## Code for Group Generator

```
public static int primit_root(int nu, int f)
 {
   int re, i, a;
   for (a = 1; a < nu - 1; a++)
   {
      for (i = 1; ; )
      {
           re = Pow(a, i, nu);
           if (re == 1)
           {
              break;
            }
            i++;
       }
       if (i == f)
          break;
   }
   return a;
 }
```

## Code for Hash and XORing Operation

```
string l = kj + u0 + K2.ToString();
byte[] sd;
byte[] fd;
sd = ASCIIEncoding.ASCII.GetBytes(l);
fd = new MD5CryptoServiceProvider().ComputeHash(sd);
int h = BitConverter.ToInt32(fd, 0);
int C13 = g1 ^ h;
```

## Code for Power Operation using Square and Multiply Method

```
public static int Pow(int e, int f, int n)
{
    int[] x = new int[32];
    x = int_to_bin(f);
    int y = sq_and_multi(e, x, n, 32);
    return y;
}
```

```
//---------- the square and multiply method --------------//

public static int sq_and_multi(int a, int[] x, int n, int nb)
{  int y = 1;
   for (int i = 0; i <= nb - 1; i++)
   {
       if (x[i] == 1)
           y = (a * y) % n;
       a = (a * a) % n;}
    return y;}
//------- to convert integer to binary ------------//

public static int[] int_to_bin(int x)
{
    int[] ans = new int[32];   int i = 0;
    while (x > 1)
    {
         if (x % 2 == 1)
             ans[i] = 1;
          else
             ans[i] = 0;
           x /= 2;
            i++;       }
       if (x == 1)
            ans[i] = 1;
        return (ans);  }
```

## Code for Sending Data

```
pno1_r = 7798
sck_r = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
 IPEndPoint localendpoint = new IPEndPoint(IPAddress.Parse(aip_r), pno1_r);
 u = textBox1.Text;
 pwd = textBox2.Text;
 try
 {
       sck_r.Connect(localendpoint);
       cx = "User is connected to authencation centre for registration ";

       //------- sending id and pwd for registration to authentication centre -------//

      text3 = u + '`' + pwd;
      data = Encoding.ASCII.GetBytes(text3);
      sck_r.Send(data);
}
```

## Code for Receiving Data

```
pno = 1235;
sck2 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
sck2.Bind(new IPEndPoint(0, pno));
sck2.Listen(1);
Socket accepted2 = sck2.Accept();
buffer2 = new byte[accepted2.SendBufferSize];
int bytesRead2 = accepted2.Receive(buffer2);
byte[] formatted2 = new byte[bytesRead2];
for (int i = 0; i < bytesRead2; i++)
{
  formatted2[i] = buffer2[i];
}
string s2 = Encoding.ASCII.GetString(formatted2);
```

## Code for Mutual Key

```
K1 = K1 + 1;
K2 = K2 + 1;
string k = r3 + d + K1.ToString();
p1 = ASCIIEncoding.ASCII.GetBytes(k);
q = new MD5CryptoServiceProvider().ComputeHash(p1);
int g1 = BitConverter.ToInt32(q, 0);
string l = kj + u0 + K2.ToString();
sd = ASCIIEncoding.ASCII.GetBytes(l);
fd = new MD5CryptoServiceProvider().ComputeHash(sd);
int h = BitConverter.ToInt32(fd, 0);

//------- Mutual Key ---------//

int C13 = g1 ^ h;
```

## Code for Session Key

```
//------ compution of sesion key at User end ------//
f3 = C17.ToString()+C14_.ToString();
w11 = ASCIIEncoding.ASCII.GetBytes(f3);
w12 = new MD5CryptoServiceProvider().ComputeHash(w11);
int C19_ = BitConverter.ToInt32(w12, 0);
if(C19_==C19)
{
   label25.Text = "User has Authencated the Server.Now,generate the mutual session key";
   Nc213 = C16  + 2;
   Ns2334 = C17 + 2;
   N8 = Nc213.ToString();
   N9 = Ns2334.ToString();
   f6 = C14_.ToString() + C14.ToString() + N8 + N9;
   w17 = ASCIIEncoding.ASCII.GetBytes(f6);
   w18 = new MD5CryptoServiceProvider().ComputeHash(w17);
   usessionkey = BitConverter.ToInt32(w18, 0);
}

//--------computation of session key at Server end -----------//

f4 = C16.ToString() + C15_.ToString();
w13 = ASCIIEncoding.ASCII.GetBytes(f4);
w14 = new MD5CryptoServiceProvider().ComputeHash(w13);
int C18_ = Convert.ToInt32(e6.ToString());
if (C18_ == C18)
{
    label19.Text = "Server has Authencated the user.Now,generate the mutual session key";
   Ns231 = C16 + 2;
   Nc2343 = C17 + 2;
   N6 = Ns231.ToString();
   N7 = Nc2343.ToString();
   f5 = C15.ToString() + C15_.ToString() + N6 + N7;
   w15 = ASCIIEncoding.ASCII.GetBytes(f5);
   w16 = new MD5CryptoServiceProvider().ComputeHash(w15);
   ssessionkey = BitConverter.ToInt32(w16, 0);
}
```

## Code for Encrypting Data

```
//------- encrypting the data for server ---------//
int q = p.Length;
int[] a = new int[q];
int i = 0;
foreach (var item in p)
{
   a[i] = item + usessionkey;
   i++;
}
//------ the data sent to server ---------//
string b = a[0].ToString();
for (int j = 1; j < q; j++)
{
   b = b + '`' + a[j].ToString();
}
//---- to show data sent by user ---------//
string b1 = a[0].ToString();
for (int j1 = 1; j1 < q; j1++)
{
   b1 = b1 + a[j1].ToString();
 }
 byte[] data = Encoding.ASCII.GetBytes(b);
```

## Code for Decrypting Data

```
//------- decrypting the received data from user ---------//
string b = Encoding.ASCII.GetString(formatted123478);
string[] s = b.Split(new char[] { '`' });
int q1 = s.Count();
int[] m = new int[q1];
int x = 0;
foreach (var item in s)
{
   m[x] = Convert.ToInt32(item) - ssessionkey;
   x++;
}
string b1 = s[0];
for (int j2 = 1; j2 < q1; j2++)
{
  b1 = b1 + s[j2];
}
char[] n = new char[q1];
for (int y = 0; y < q1; y++)
{
   n[y] = Convert.ToChar(m[y]);
}
string c = n[0].ToString();
for (int j1 = 1; j1 < q1; j1++)
{
   c = c + n[j1].ToString();
}
string data = c;
```

## Code for Diffie-Hellman Key Exchange

```
//--------- At user end ---------//
int aq = r.Next(p); //generation of aq.
int C1 = Pow(g, aq, p);
// user sends C1 to authentication center over public network and in return receives C5 to computes value
of K1.
 int C5 = Convert.ToInt32(strarr[0]);
 int K1 = Pow(C5, aq, p);

//---------- At authentication center end --------------//
int cq = r.Next(p);  //generation of cq.
int C5 = Pow(g, cq, p);
// authentication center C5 to user and computes K1 using the received value of C1.
int K1 = Pow(C1, cq, p);
```