CHAPTER 1

1. Introduction

In modern electronic distribution networks, data communication between remote parties is an important aspect of our living. An explosive growth in the amount of sensitive data exchanged over the internet have seen with the rapid use of various applications like online banking, stock trading, e-health and corporate remote access. So, the defending of data or information and information systems from unauthorized access, use, disclosure, disruption, modification, recording or destruction is essential. Cryptography is the study of techniques for ensuring the secrecy and/or authenticity of information and makes the computer systems secure enforcing the confidentiality and integrity [24]. A symmetric key cryptosystem where only one key is used for both encryption and decryption and an asymmetric key cryptosystem where two different keys are used, one for encryption and another for decryption, constitute the modern cryptographic system of crypto systems [31, 39]. Even though private key cryptography approaches are more efficient in terms of length and number of keys used than public key cryptography approaches, they are not dominant player in securing communication systems given their difficulty of providing secure key management [14, 31]. An asymmetric or public key cryptosystem is widely used in network security and internet communications for authentication and encryption.

Authentication is the process of confirming the truth of an attribute of a datum or entity and it involves verifying the validity of at least one form of identification. Cryptosystem is an important technique to identify the authenticity for protecting the confidential and sensitive data in mobile networks [28]. Authentication system supports the privacy protection and ensures that entities verify and validate one another before disclosing any secrete information. By allowing access to services and infrastructures for authorized entities and denying unauthorized entities access to sensitive data, authentication ensures confidentiality, data integrity and access control. In case of mobile networks, air is the communication channel for data transmission so there are higher possibilities for information snoop from nodes by un-authorized base stations that are pretending as the valid one. To have reliable proper security over the mobile networks it is necessary to provide authentication as one of the major security measures [28]. Due to the high

chance that the requesting station in mobile network would be an adversary so, it is very essential to authenticate the base station before transfer of secrete information. Otherwise the pretending one can gain unauthorized access to resources and sensitive information from nodes.
In this context, this study includes an empirical analysis regarding computational aspects of existing direct and indirect authentication mechanism so as to suggest an efficient and effectiveness authentication system in mobile networks.

1.1 Objective

- To implement and analyze the direct and indirect authentication techniques in mobile network containing resource constrained devices, using different techniques of cryptography in order to gain the efficient measure for authentication protocols.

1.2 Thesis Organization

After introducing the cryptographic systems, application of these systems in an authentication mechanism for secure access of information in insecure environment and objectives of secure communication in this chapter, the rest of the material in this study is organized into subsequent five chapters.

Chapter 2 includes background study required for this dissertation. In this chapter, the problem or significant challenge during authentication in case of mobile networks is given, problem statement is formulated and how this study response those issues related to authentication in mobile networks is clarified.

In Chapter 3, the previous works related to this study are described in detail under the literature review. The chapter covers various direct and indirect authentication systems using public key cryptography such as ECC, RSA and some other authentication schemes in detail.

Chapter 4 contains an implementation overview of direct and indirect authentication schemes that include ECC based direct authentication, ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman authentication mechanism and

Chang-Cheng's authentication mechanism in mobile network using J2ME™. The implementation details with sample codes are included within this chapter.

Chapter 5 includes the empirical analysis of authentication times for computational cost and performance evaluation of ECC based direct authentication, ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman authentication mechanism and Chang-Cheng's authentication mechanism. The result of the study is shown in tables as well as in graphs.

Finally, the concluding remarks and further recommendations are outlined in chapter 6.

CHAPTER 2

2. Problem Definition and Background Study

Authentication, access control and security are vital features that must be present in any communications network. The need for these features is more in the case of wireless mobile communications than in wired communications because of the ubiquitous shared nature of the wireless medium in mobile network [13]. In current electronic communication system sharing of data and information between various parties from various locations through insecure channel is quite general. Generally, majority of the fields such as banking sectors, health organizations, government offices and different business organizations etc use common insecure communication channel for data transmission between them. In this type of communication systems an unwanted entities may get access to resources, information systems and are able to get sensitive information [10, 13, 30]. So, before sharing the secure and sensitive data, information and providing access to various resources, the identification of the intended or claimed entities is one of the major essential factors for any secure communication system.

2.1 Problem Definition

One of the biggest problems faced in current internet business is the thorny issue of trust and security. The vast majority of internet users are concerned about the safety of their secret information and personal details from being accessed by unauthorized entities. Authentication is a must in order to achieve the necessary trust [28, 39]. To ensure the integrity and safety of the information, it is important to identify with whom we are dealing, and that the data we are receiving is trustworthy.

For any networking environment, a secure communication is an important aspect during the data transmission and is especially a significant challenge in case of mobile networks since they are deployed in un-trusted environments [26, 32]. In case of mobile networks, there is unlimited mobility and information access occurs frequently between unacquainted nodes and servers. Keeping the privacy and security of data in such un-trusted mobile network is a big concern. There is high chance of eaves-dropping in mobile networks and the un-authenticated access of

base station on nodes is the root cause of such eavesdropping. Authentication is one of the major security issues affecting the wired and the wireless network community. Authentication provides initial level of security by identifying the valid users or nodes. So, selection of an efficient and effectiveness authentication system is an important factor in mobile networks where handheld devices are resource constrained in nature. So, it is desirable to use most efficient authentication mechanism among various authentication schemes. Thus in this context, this study will include analysis of various direct and indirect authentication mechanisms i.e. ECC based direct authentication, ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman Authentication Mechanism and Chang-Cheng's Authentication Mechanism so as to suggest an efficient authentication mechanism for mobile networks.

2.2 Background Study

In any research and study require the basic terms and terminologies related to that study. In this context the basic background study related to the authentication are outlined in the following sections;

2.2.1 Cryptographic System and Secure Communication

Earlier cryptosystems were simply designed to encrypt and decrypt a message by their respective senders and receivers so that the message is incomprehensible to an intruder. However nowadays cryptography considers the study and practices of not only message confidentiality i.e. encryption but also for authentication, integrity checking, key management, etc [39]. Modern cryptography concerns itself with the following objectives of secure communications between the two communicating parties [35, 39]:

Data confidentiality: Data confidentiality refers to the protection of data from unauthorized disclosure. This means the information cannot be understood by anyone for whom it was unintended.

Data integrity: The assurance that data received are exactly the same as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay). This assures that the

information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected.

Authentication: The sender and receiver can confirm each other's identity and the origin/destination of the information.

- **Peer Entity Authentication:** The assurance that the communicating entity is the one that it claims to be. It is used in association with a logical connection to provide confidence in the identity of the entities connected.
- **Data Origin Authentication:** The assurance that the source of received data is authentic. In a connectionless transfer, provides assurance that the source of received data is as claimed.

Non-repudiation: The protection against denial, by one of the entities in the communication from having participated in the communication. The creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information.

Among all above objectives, the authentication plays an important role in security measure that is the initial need before sharing of secrete and confidential data between unidentified entities. Authentication systems designed using modern cryptographic techniques ensure the prior level of security [39].

Generally, various authentication schemes use different cryptographic systems to enhance the proper security to these schemes. The modern cryptographic system or cryptosystems that may be used in authentication are broadly classified into following two types by depending upon the number of keys used [14, 36, 39].

- Private key cryptography/Symmetric-key systems, where a single secret key is used by both the sender and recipient for both encryption and decryption. Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way.
- Public key cryptography/public-key systems, where two different keys are used; one for encryption and another for decryption. One key is kept secret and the other is made

public. Public-key cryptosystems avoid secure key exchange problem because the public key can be distributed in a non-secure way, and the private key is never transmitted.

The major advantage of the symmetric-key cryptography is its high efficiency but the reason of existing significant drawbacks or limitations mentioned below by authors in [14, 31, 39], this type of cryptosystem is not used in the new generation of authentication systems. The limitations are as follows:

- **Key Distribution Problem:** Two parties wanting to communicate need to set up a shared secret key in advance before they can start communication over an insecure channel. A secure channel for selecting a key over public media (like Internet) may not be available.

- **Key Management Problem:** In a network of n users, every pair of users must share a secret key leading to a total of n*(n-1)/2 keys. If n is large, then the number of keys become unmanageable.

- **Authentication Problem:** In electronic communication systems an equivalent of a signature is needed for trust and secure communication. Symmetric cryptosystems are unable to provide this feature. When there is a conflict between sender and receiver, it is impossible to decide who is right. Any message made by one of them could also have been made by the other ones [14]. In public key cryptography a digital signature is used for authentication and that allows the receiver of a message to convince any third-party that the message in fact originated from the sender. However, in a private key cryptosystem, Alice and Bob both have the same keys for encryption and decryption and thus Bob cannot convince a third party that a message he received from Alice in fact originated from Alice.

2.2.2 Authentication Using Public Key Cryptography

Public-key cryptosystems require only the communication entities exchange keying material that is authentic but not secret. The private key is kept secret and public key is used as public by the

entity. This concept was invented by Whitfield Diffie and Martin Hellman in 1976 and independently by Ralph Merkle in 1978 [14, 28, 31]. In most of the authentication schemes that are used in the current distributed networking environments, a public key cryptography is used where only public key is shared and no need to share secrete key between the communicating parties. Public-key cryptography provides solutions to the problems of symmetric-key cryptography, specifically the key distribution, key management, the provision of non-repudiation and authentication problem.

According to the author in [39], the authentication system uses a public key or asymmetric cryptosystem during generation of authentication code because of the conditions it meets as:

- It must be computationally easy to encrypt or decrypt a message given the appropriate key.
- It must be computationally infeasible to derive the private key from the public key.
- It must be computationally infeasible for an adversary, knowing the public key and a cipher text, to recover the original message.

The security level of authentication system is derived from the hardness of the mathematical problem implemented in the public-key cryptosystem that is used in the specific authentication system. The common mathematical problems that are related to various public key crypto systems can be Integer Factorization Problem, Discrete Logarithm Problem, Elliptic Curve Discrete Logarithm Problem [21, 39].

2.2.3 RSA Public Key Cryptography

RSA is one of the first practicable public-key cryptosystems and is used as a digital signature in authentication. The security of RSA is based on the intractability of the integer factorization problem. According to T. Struk in [36], the problem states: find two different large prime numbers p and q having product of multiplication N = p*q. In this type of cryptosystem the encryption key is public and differs from the decryption key which is kept secret [13, 35, 39]. When two parties want to communicate securely with one another using the RSA cryptosystem scheme, require the generation of public/private key pairs. Sender can send a message to anyone by enciphering the message with the receiver's public key; however only the destined receiver

can decipher the message using his private key. To obtain the public keys anyone should use the valid certificate. A certificate is provided by a certificate authority, such as a government agency or a financial institution that is trusted by the user community. A user can present his/her public key to the authority in a secure manner, and obtain a certificate. The user can then publish this certificate so that the other user needing to communicate can obtain its public key [31, 39]. A user with invalid certificate will not be able to get the public key of any other users. At the receiver end the signature code received from the sender is decrypted using public key of sender and verifies for authentication [31, 35].

2.2.4 Elliptic Curve Cryptography

Elliptic curve cryptography is a new approach to public key cryptography based on the algebraic structure of elliptic curves over finite fields [5, 31, 39]. Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller and Neil Koblitz as an alternative mechanism for implementing public-key cryptography [12, 13]. The entire security of elliptic curve cryptography depends on the ability to compute a point multiplication and the inability to compute the multiplicand given the original and product points. According to the key size, bandwidth requirements of ECC is said to provide greater efficiency than either integer factorization systems or discrete logarithm systems. This means that higher speeds, lower power consumptions and reduced code size are the advantages of ECC. That is why ECC is widely used in the variety of authentication systems used in various applications. Furthermore, the hardness of the underlying mathematical problem ECDLP appeals ECC public-key cryptosystem for applications demanding high security [19, 20].

The security of ECC is based on the mathematical problem called Elliptic Curve Discrete Logarithm Problem (ECPLD). This states that a given an elliptic curve E defined over a finite field Fq, points P and Q lying on the curve find an integer k such that Q = kP [14, 28, 29]. The problem is computationally intractable for large values of k. This computational problem becomes harder with the size of domain parameters given in [12].The primary benefit promised by ECC is a smaller key size, reducing storage and transmission requirements, i.e. that an elliptic curve group could provide the same level of security afforded by an RSA-based system with a large modulus and correspondingly larger key. Solving ECDLP takes more time than the integer

factorization and DLP when the same key sizes are used. This advantage allows ECC to achieve the same level of security with smaller key sizes and higher computational efficiency [35].

After the invention of ECC the issue of implementing public key cryptography on mobile computing devices has been resolved [20, 31]. These beneficial features of ECC provide faster computations and lower power consumption besides memory and bandwidth savings and so ECC is used popularly in most of the authentication systems [35].

2.2.4.1 Group Law of EC Group of Points

Let E be an elliptic curve and the set E_p (a, b) satisfies the required properties of becoming a finite abelian group with addition operation denoted by +. The addition is performed according to a rule called chord-tangent-rule [18]. Elliptic curve cryptosystems are constructed on abelian groups and various operations defined on the elliptic curve points are as follows:

For all points P, Q, R ε E_p (a, b) with O, the point at infinity, serving as the identity element,

(1) Identity: $P + O = O+P = P$

(2) Negative or Inverse: If $P = (x_P, y_P)$ then $-P = (x_P, -y_P)$ and $P + (-P) = O$

(3) Point Addition: If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ with $P \neq -Q$, then $P + Q = R = (x_R, y_R)$ is the mirror image of the point of intersection of the chord through P and Q and the curve and is determined by following rule;

$x_R = (\lambda^2 - x_P - x_Q) \bmod p$

$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$

where $\lambda = ((y_Q - y_P) / (x_Q - x_P)) \bmod p$

(4) Point Doubling: If $P = (x_P, y_P)$ where $P \neq -P$ then $2P = R = (x_R, y_R)$ is the mirror image of the point of intersection of the tangent through point P and the curve. The result point is determined by following rule;

$x_R = (\lambda^2 - x_P - x_Q) \bmod p$

$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$

and $\lambda = ((3x_P^2 + a) / 2y_P) \bmod p$ if $P = Q$.

(5) Associative and Commutative properties can be verified easily using property (3).

The geometric addition of elliptic curve points is shown in Figure: 2.3 and Geometric doubling of elliptic curve point, $2P=R$ is depicted in Figure: 2.4.

.

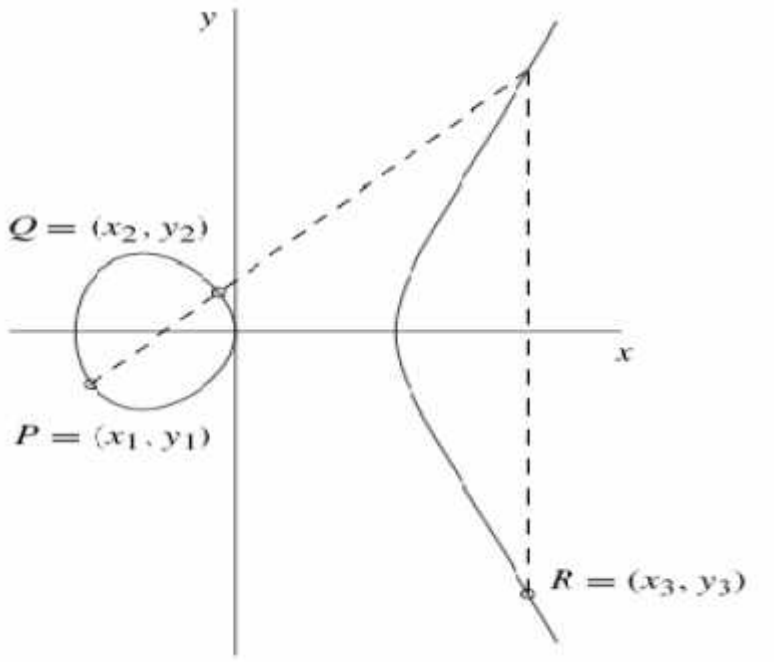

Figure 2.1 Geometric addition of elliptic curve points, $P+Q=R$. [18]

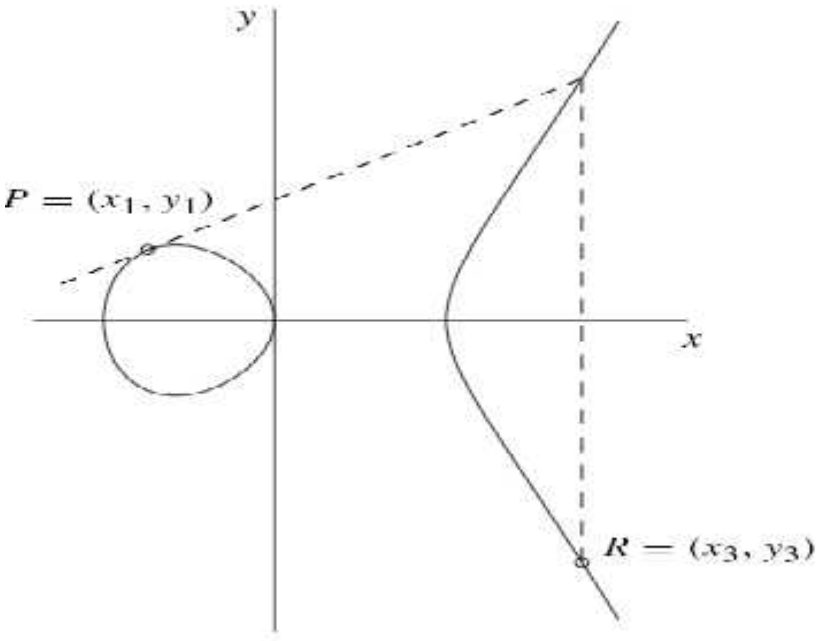

Figure 2.2 Geometric doubling of elliptic curve point, $2P=R$ [18]

2.2.4.2 Multiplication over an EC Group

Point multiplication by a scalar is another basic operation in ECC. Suppose P be a point on an elliptic curve E that is multiplied by an integer n i.e. Q = n*P. The multiplication over an elliptic curve group $E_p(a, b)$ is the equivalent operation of the modular exponentiation in RSA [35]. The multiplication over an elliptic curve group $E_p(a, b)$ by a scalar is a series of additions and doubling of points. As mentioned in [36] the multiplication n*P is defined as repeated addition; e.g. n * P = P + P + …+ P (n times). It is used in elliptic curve cryptography (ECC) as a way of producing a trap door function. There are many algorithms performing the operation and many researchers work on finding more and more efficient algorithms. Point multiplication is the dominant operation in ECC cryptographic schemes. This is the operation which is the key to the use of elliptic curves for asymmetric cryptography -the critical operation which is itself fairly simple, but whose inverse (the elliptic curve discrete logarithm) is very difficult. To get the public key in ECC based authentication, the multiplication of base point by private key is performed.

2.2.5 Finite Fields: GF (p)

In cryptography finite fields have played a crucial role and become increasingly important in a number of cryptographic systems. A field F containing a finite set of objects i.e. field elements together with the description of two operations addition and multiplication performed on pair of field elements is called finite field [11]. The number of elements in filed F represents the order of a finite field F. Let p be a prime number where p>0 then the set {0, 1, …, p-1} under the operations of addition and multiplication modulo p is a finite field of order p [39]. In this case p is the order of this finite field and also called the Galois field of order p, in honor of the mathematician who first studied finite fields. A finite field F having order p is indicated by various notations including GF(p), Fp, Z/pZ and Zp, where Z denotes the set of integers [11]. Elliptic curve cryptography makes the use of elliptic curves in which the variable and coefficient are all restricted to elements of a finite field.

Let GF(p) is the prime finite field containing p elements. A set of integers {0, 1, …, p-1} represent the overall possible elements contained within the field GF(p) under the operation addition and multiplication. Various operations involved in finite fields are as follows:

Addition: If a, b e GF(p), then a + b = r e GF(p), where r e [0, p - 1] is the remainder when the integer a + b is divided by p. This is known as addition modulo p and written a + b = r (mod p).
Multiplication: If a, b e GF(p), then a * b = s e GF(p), where s e [0, p - 1] is the remainder when the integer a * b is divided by p. This is known as multiplication modulo p and written a * b = s (mod p).

For Galois field GF(p), the integer 0 (zero) represents the additive identity or zero element and the multiplicative identity is the integer 1 [39]. The subtraction and division operation in field elements are defined as additive inverse and multiplicative inverse respectively. As described in [11] the additive inverse and multiplicative inverse for the field elements are defined as follows:
Additive inverse: For a field element a e GF(p), the additive inverse of a is denoted by (-a) and is defined as the unique solution to the equation a + x = 0 (mod p).
Multiplicative inverse: For a field element a e GF(p),where a does not equal to 0, then the multiplicative inverse a-1 of a in GF(p) is the unique solution to the equation a * x = 1 (mod p).

2.2.6 Mobile Network

A mobile network can be defined as an autonomous collection of mobile nodes which communicates over a wireless link. The communication is freed from the location constraints of the stationary wire line infrastructure. Due to the mobile connectivity, the user will be permitted to access the information anytime, anywhere [28]. A mobile network has several advantages which include: increased capacity, reduced power usage and better coverage. These advantages make mobile networking pertinent to various real time applications. Air is the communication channel in case of mobile network which provides plenty of possibilities for information snoop from nodes by un-authorized base stations that are pretending as the valid one, thus being the reason for the several security issues.

2.2.7 Authentication

Authentication is the existence of some method for ensuring that the entity to which we are talking to is who it claims to be. In other words an authentication is the act of establishing or confirming something or someone as authentic, that is, that claims made by or about the thing are true [2, 9, 25]. Authentication is one of the major security issues that affect the wired and

wireless mobile networks [16, 22]. Protection against replay attacks; confidentiality, resistance against man-in-the-middle attacks etc. are several requirements for authentication that are employed by any authentication schemes include all the above requirements [4, 5, 9]. Therefore, it is necessary to attain authenticity which is an essential prerequisite achieved by employing cryptographic systems in most applications where security matters. Authentication is an initial process to authorize a mobile terminal for communication through secret credentials so as to provide security services in wireless mobile networks [9, 15, 30]. The communicating two parties must authenticate one another. This two-way authentication is called the mutual authentication. During this process the client must be authenticated by the server and the user also need to authenticate the server to which it is communication [9, 29].

Every authentication system consists of five components that are required for overall authentication process [38] and they are as follows:

1. The set A of authentication information is the set of specific information with which entities prove their identities.
2. The set C of complementary information is the set of information that the system stores and uses to validate the authentication information. The set of complementary information may contain more, or fewer, elements than A, depending on the nature of the complementation function.
3. The set F of complementation functions that generate the complementary information from the authentication information. That is, for $f \in F$, $f: A \rightarrow C$.
4. The set L of authentication functions that verify identity. That is, for $l \in L$, $l: A \times C \rightarrow \{true, false\}$.
5. The set S of selection functions that enable an entity to create or alter the authentication and complementary information.

Authentication system supports the privacy protection and ensures that entities verify and validate one another before disclosing any secrete information. In modern cryptography authentication system can be broadly categorized as Direct and Indirect authentication [3, 5, 10, 28] .

2.2.7.1 Direct Authentication

In direct authentication, the base station or user node directly communicates with the main server during authentication process. In this type of authentication, the communicating two parties use pre-shared symmetric or asymmetric keys for verifying each other and the flow of data between them [22]. When requesting information by user node to the main server, the user node needs to be authenticated by main server. If the base station is valid then only the main server has to send the requesting information to user node otherwise it has to send a warning message. Direct authentication requires the presentation of user credentials. The service uses these credentials to authenticate the requested user node. The direct authentication system is commonly used in peer to peer systems.

2.2.7.2 Indirect Authentication

In indirect authentication system, two servers are involved to perform authentication process, one is main server and another is authentication server. In other words indirect authentication means that entities use intermediary entities to authenticate each other [2, 5]. A trusted third party is involved in the indirect authentication protocol. Certification of the two communication parties is the responsibility of this third party such as certification Authority. In indirect authentication, the authentication server is commonly used to perform the authentication [3]. Indirect authentication is the most widely used protocol in case of wired and wireless network in distributed systems.

2.2.8 Authentication Factors

Generally, a term strong user authentication is used to describe any authentication process that increases the probability of the correct verification of the identity of base station. To meet the requirements of strong user authentication, long complicated passwords or combination of two or more authentication factors are used to accomplish it. According to the authors in [11, 15, 29] there are three authentication factors as:

- Knowledge factor – the user has to present the information known by him/her, i.e. some secret information such as, a password, PIN.
- Possession factor –the user has to present something possessed by him/her, i.e. some physical object for instance- tokens or keys.

• Being factor – In contrast to the former two factors, the user has to present something that is a part of him/her, for example- physical parameters. i.e., some biometric data e.g., fingerprint, iris pattern.

Use of passwords is the most common case and is not really good choice because passwords are short and easy to break. Currently some other more secure methods are used where the public key cryptography, challenge response schemes, symmetric encryption, etc are included. Mainly the method involved in the generation of secured key pair determines the effectiveness of any authentication protocol. The authentication schemes that are used in this study use the public key cryptography during key pair generation.

2.2.9 Authentication Functions

Any authentication system or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator that means a value to be used to authenticate a message. This lower level function is then used as a primitive in a higher level authentication protocol that enables a receiver to verify the authenticity of a message. Among various authentication functions that are used to authenticate the message sent by the user node, hash functions such as SHA-1, SHA-256 etc. are used in this study.

2.2.9.1 Hash Functions

Hashing functions or hash functions are used to condense an arbitrary length message to a fixed size, usually for subsequent signature by a digital signature algorithm. A cryptographic hash function provides message integrity and authentication. A good cryptographic hash function h should have the following properties:

- h should be computed on the entire message
- h should destroy all homomorphic structures in the underlying public key cryptosystem.
- h should be a one-way function so that messages are not disclosed by their signatures
- it should be computationally infeasible given a message and its hash value to compute another message with the same hash value

2.2.10 Certificate Distribution

Certificates are utilized when it is necessary to exchange public keys with someone else. For small groups of people who wish to communicate securely, it is easy to manually exchange diskettes or emails contain each owner’s public key. This is manual public key distribution and it is practical only to a certain point. Beyond that point, it is necessary to put systems into place that can provide the necessary security, storage and exchange mechanisms so coworkers, business partners, or strangers could communicate if need be. These can come in the form of storage-only repositories called Certificate Servers or more structured systems that provide additional key management features and are called Public Key Infrastructures (PKIs) [26]. A Certification Authority (CA), an authorized organization, provides facilities like certificate storage, certificate management and issuing certificates to the end users. As part of a public key infrastructure (PKI), a CA checks with a registration authority (RA) to verify information provided by the requestor of a digital certificate. If the RA verifies the requestor's information, the CA can then issue a certificate.

CHAPTER 3

3. Literature Review

Several research works have been done in the past and proposed a number of public-key cryptographic systems that are deployed in currently used various authentication schemes in mobile network [37]. Some of the researchers suggest RSA based authentication systems where the security of the system is based on the intractability of the integer factorization problem [15, 34]. Some research works have been done in authentication using optimized versions of RSA public key cryptography. Armando Fox and Steven Gribble, have described an implemented indirect protocol called Charon, which provides authentication and secure communication to clients by leveraging the strong protocol and deployed infrastructure of Kerberos IV [2n].

J. H. Lee and D. H. Lee designed a more efficient remote authenticated key agreement authentication scheme for multi-server in 2008 [17]. Their proposed method adopts only light operations, such as hash function and exclusive-OR. However, there is a design weakness in Lee and Lee's scheme, which can suffer the forgery attack. Some research works have been done using ECC for direct and indirect authentication systems. V. Vijayalakshmi et al. [36] have proposed an authentication technique which makes use of Elliptic Curve Cryptography (ECC) along with the TOA positioning scheme was implemented to solve the problem of insecurity in mobile networks. From research ECC got excellent enhanced features which include smaller key size, lesser bandwidth, higher computational capability and lesser hardware [5, 34].

In December, 2012, C. T. Li, C. C. Lee, H. Mei and C. H. Yang proposed an improved version of smart card based password authentication mechanism in multi-server mobile networks. Compared with other related protocols, performance analysis shows that this proposed mechanism is still cost-efficient for the real application in multi-server environments [9n]. Most of the recent research works on direct and indirect authentication systems have been focused mainly upon the nature and requirements of the applications where the authentication is required using the public key cryptography [5, 9, 16].

3.1 ECC Based Authentication

Elliptic curve cryptography plays an important role in authentication and encryption protocols [26, 34, 38]. Elliptic curve cryptography is an approach to public key cryptography based on the algebraic structure of elliptic curves over finite fields [5]. ECC has been adopted in a wide variety of applications from digital certificates in web server authentication to embedded processors in wearable devices [7, 8, 13, 33, 34].

3.1.1 ECC Based Direct Authentication

In case of ECC based direct authentication, the user node initiates message transfer by requesting to main server with requesting code R_c. Before sharing the confidential and secrete data using the resources of the server node, first of all the requesting base station or user node should be an authorized one. So, the main server response for the request and then it verifies the authentication of the requested base station. If the base station is an authorized one then it gets the access right to the server resources otherwise it is declared as unauthorized base station and server transfers warning message to the user node.

In ECC based direct authentication process, the key generation is the initial phase that is influencing wholly by the elliptic curve cryptography. The mathematical derivation for the generation of key pair is as follows:

An equation of an elliptic curve E defined over a prime finite field F_p can be written as follows

$$y^2 = x^3 + ax + b$$

Where $a,b \in F_p$ should satisfy,

$$4a^3 + 27b^3 \neq 0 (\text{mod } p)$$

Let ∞ denotes the point at infinity which is said to be on the curve. The $E(F_p)$ denotes the set of all the points on the elliptical curve E. Let a point P in $E(F_p)$ and let another point B that has prime order m . Then B generates the cyclic subgroup of $E(F_p)$ as:

$$< B >= \{\infty, B, 2B, 3B, ...(m-1)B\}$$

The equation of the elliptic curve E , the prime p, and the point B and its order m , are the public domain parameters. A private key is an integer K_s that is selected uniformly at random from the interval $[1, m-1]$, and the corresponding public key is given by

$K_p = K_s B$

The factors B , K_p and K_s are found out with the help of the properties of elliptical curve. K_p , the public key and K_s, the private key belongs to the base station and B is the generating point from the points of elliptical curve [28].

The overall process is depicted as a schematic view of the direct authentication protocol by the following signal flow diagram:

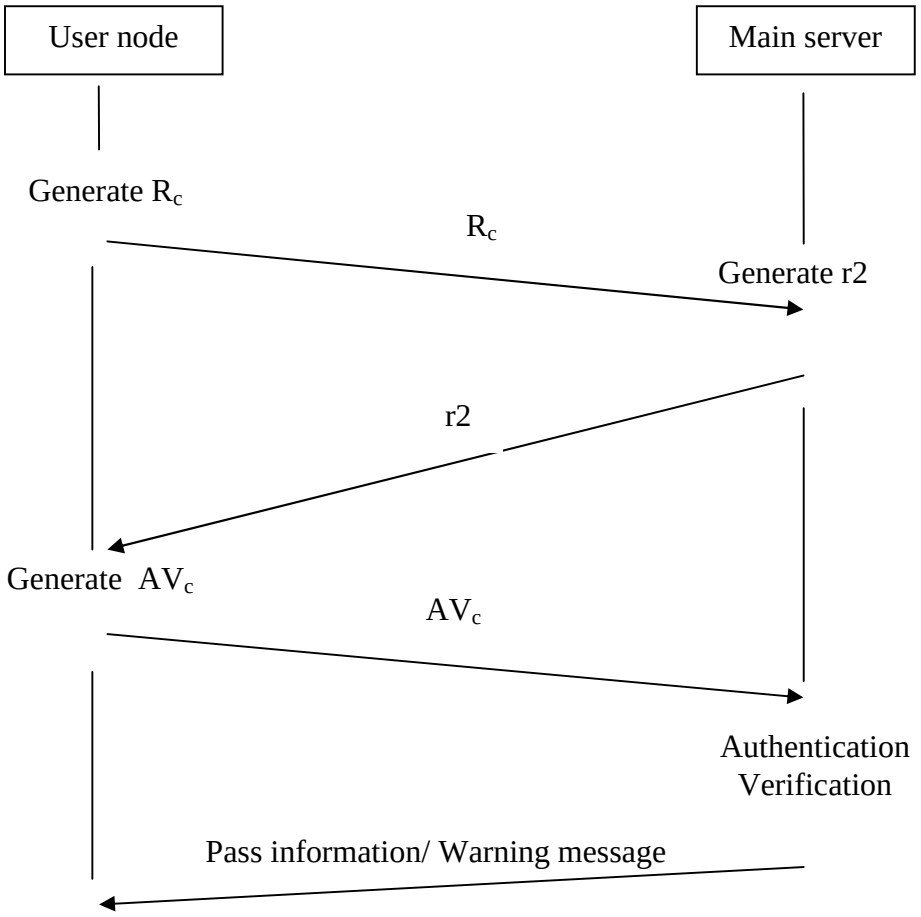

Figure 3.1: Protocol flow of ECC based direct authentication

First of all, the requesting base station generates a random number r1. Then it will calculate the requesting code R_c as follows:

$R_c = r1 * B$

The base station then sent this value R_c to node as request and the node will then generate a random number r2 and send it again to base station. The base station will generate authentication-verifying code AV_c as a response. AV_c is derived as follows,

$$AV_c = r1 + (r2 * K_s)$$

This AV_c is sent to the node for its authentication verification and the code verification is given below

$$(AV_c * B) - (r2 * K_p) = R_c$$

If the above condition is satisfied, the node will come for a conclusion that the base station is a valid one otherwise a warning about its un- authentication will be given to the corresponding base station.

3.1.2 ECC Based Indirect Authentication

The ECC plays an important role in the generation of key pairs for Authentication server as well as Main server [8, 29]. Various signals have been transferred between the user node, Authentication Server and the Main Server. During key generation ECC is used. The generation of key pair can be mathematically written as follows:

An elliptic curve E over F_p can be written in the form of equation as follows:

$$y^2 = x^3 + ax + b$$

where $a\ b \in F_p$, and the following relation should be satisfied,

$$4a^3 + 27b^2 \neq 0 (\text{mod } p)$$

The point at infinity denoted by ∞ and is said to be on the curve. The set of all the points on the Elliptical curve E is denoted by $E(F_p)$.

Let $E(F_p)$ contains a point P . Let another point B that has prime order m, the cyclic subgroup of $E(F_p)$ generated by B is

$$< B >= \{\infty, B, 2B, 3B, ...(m-1)B\}$$

The prime p, the equation of the elliptic curve E , and the point B and its order m are the public domain parameters . An integer Ks is a private key is that is selected randomly from the interval $[1, m-1]$.

The private key chosen for authentication server within the interval [1,m −1] is K_s (A.S.) and for Main server the private key is K_s (M.S.) chosen from the same interval.

$$K_p \text{ (A.S.)} = B * K_s \text{ (A.S.)}$$

where K_p (A.S.) is the public key of the Authentication server

K_p (M.S.) = B*K_s(M.S.)

where K_s (M.S.) is the private key of the Main server. The properties of Elliptical curve aid in the eventual identification of the protocols parameters B , K_s (A.S.) , K_s (M.S.) ,K_p (A.S.) , and K_p (M.S.) .

The protocol flow of ECC based indirect authentication is depicted in the following diagram:

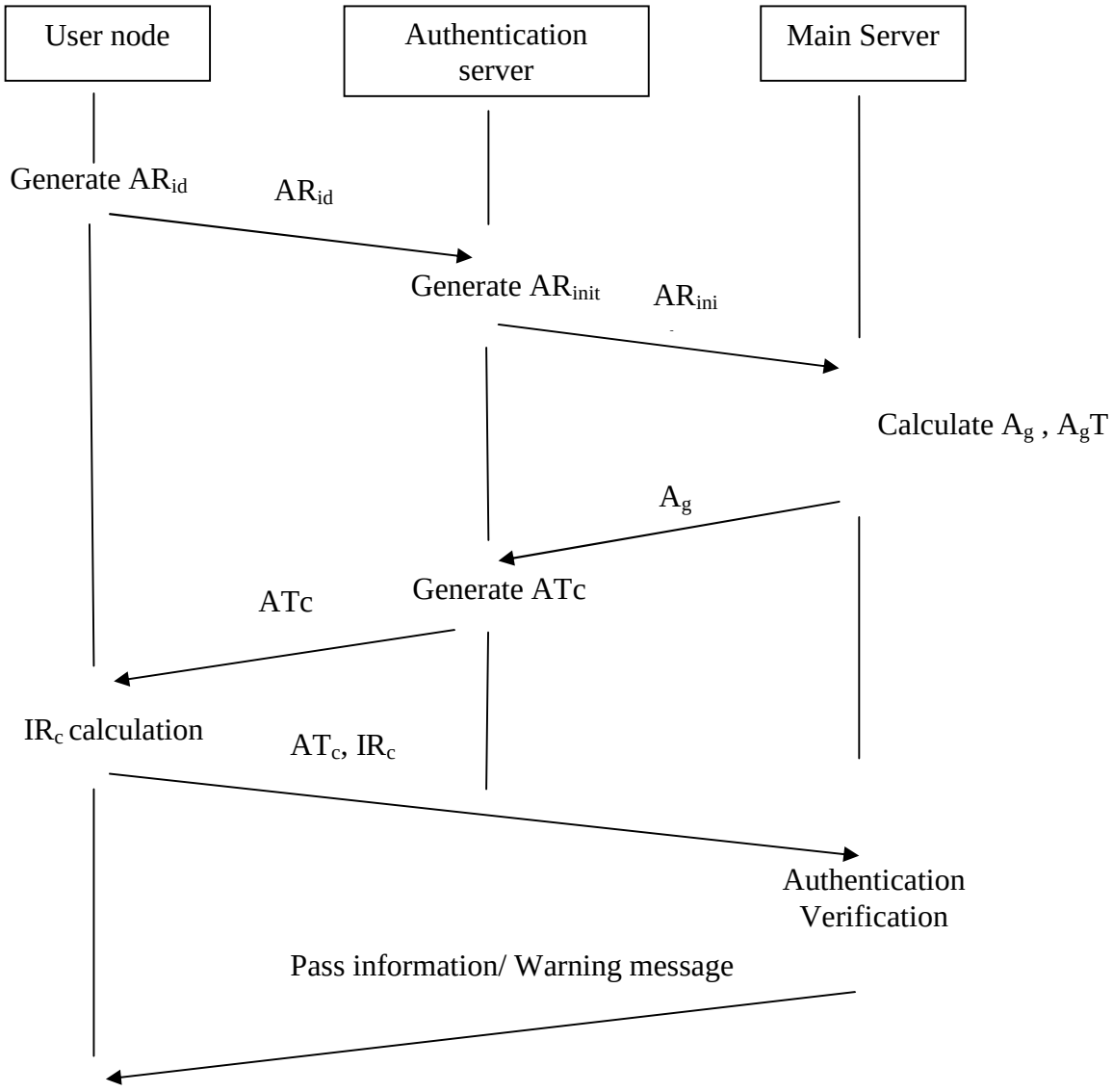

Figure 3.2 Protocol flow of ECC based indirect authentication

The overall procedures involved during authentication of the user node in ECC based indirect authentication are described as follows:

- First of all the User generates AR_{id}, which is a random number, and sends it to Authentication Server
- Authentication Server also generates $AR_{int,}$ another random number, and sends it to Main Server.
- Main Server generates Ack_{rand} and calculates A_g and A_gT by using the random number AR_{int} generated by authentication server as follows,
  $A_g = B[AR_{int} * K_s(M.S.)]$
  $A_gT = [Ack_{rand} * K_p(A.S.) – AR_{int} * K_p(M.S.)]$
- After then the Main Server sends A_g to Authentication server and keeps A_gT secrete for future use in authentication process.
- Authentication Server calculates AT_c as given below and sends it to user node
  $AT_c = [AR_{id} + Ack_{rand} * K_s(A.S.)]B – A_g$
- User node calculates IR_c as given below and sends *IRc* and *ATc* to Authentication Server,
  $IRc = ARid * B$
- Finally the Main Server performs authentication using the parameters *ATc* and *IRc* that are received from the user node.
  $ATc – A_gT = IRc$

If the above relation, ATc – A_gT = IRc, is satisfied then it proves that the certification code is efficient in identifying the valid user node and it will confirm to neglect for sharing the information to any un-authorized node.

3.2 RSA Based Authentication

In RSA system, two parties that want to communicate with each other securely using the RSA cryptosystem scheme, require the generation of public/private key pairs. Anyone can send a message to someone in network by encrypting the message with the receiver's public key; however only the destined receiver can decrypt the message using his private key. For obtaining one's public key, certificates are used.

A certificate is provided by a certificate authority which can be a trusted server, or the communicating parties can exchange their credential certificates themselves, based on the approach of indirect or direct authentication. A user can present his/her public key to the authority in a secure manner, and obtain a certificate. The user can then publish this certificate so that the other user needing to communicate can obtain its public key [21, 39]. Using RSA as an authentication mechanism, the sender sends an encrypted authentication message with the receiver's public key. Along with that the sender also creates an encrypted signature for authentication. The authentication token i.e. the signature, is generated by encrypting the authentication message once with the private key of sender and then with the public key of the receiver. At the recipient end, the receiver first decrypts the encrypted authentication message with his/her private key. Along with that the receiver also decrypts the signature with the private key of own receiver and then with the public key of the sender.

Finally, the decoded signature is compared with the decrypted authentication token for its validity. Based on the use of certificate authority, the RSA authentication scheme can also be either direct or indirect.

3.2.1 RSA Based Direct Authentication

In case of RSA based direct authentication, the requesting user node directly communicates with the main node. Before accessing the resources and services the user node must be authenticated. To complete the authentication process the requesting node creates a signature message 'S' that is encrypted by public key (E_b,N_b) of another main node whose resources are willing to access from the user node. The signature code generated in this way is then transferred directly to the main server for authentication. The main node then verifies the signature message received from requesting node for authentication using public key (E_a,N_a) of requesting node [31, 39]. After the completion of authentication process, if the requesting node found as authentic then main server convey positive message otherwise it sends warning message. The signature 'S' at requesting node is calculated as;

1) $A \leftarrow h(M)^{Eb} \pmod{N_b}$, where M is the message to be signed.
2) $S \leftarrow A^{Da} \bmod N_a$
3) The signature S is then transferred to main node.

The requesting node is authenticated in main server by performing the following calculations;

1) $B \leftarrow S^{Ea} \bmod N_a$
2) $Q \leftarrow B^{Db} \bmod N_b$
3) Calculate h(M) , if Q = h(M), then the requesting node is authenticated as valid node otherwise warning message is sent to user node.

The overall authentication process of RSA based direct authentication scheme can be depicted in the following diagram:

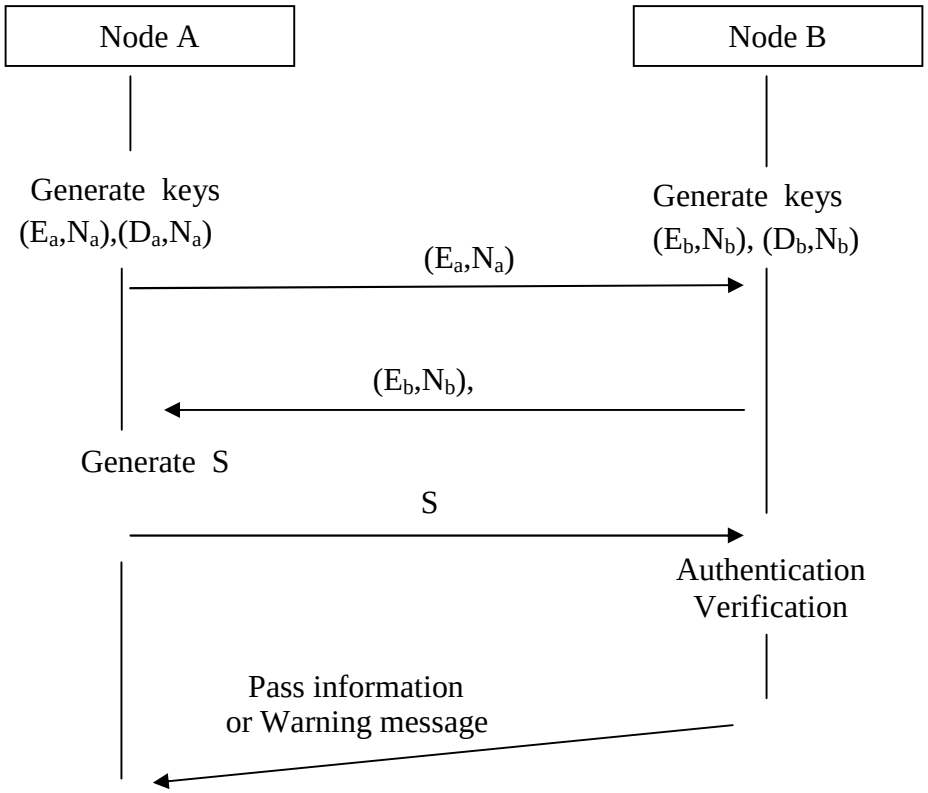

Figure 3.3 Protocol flow of RSA based direct authentication

3.2.2 RSA Based Indirect Authentication

In this approach a middle server is used as an intermediate authentication node in between the two communicating parties. First of all, these two communicating parties share public keys to each other [31]. Then the requesting node sends authentication message 'S' to middle server where one level of authentication completes and then it sends authentication code to main server. The main server then verifies authentication code for authentication by using public key (Ea,Na)

and sends message to access resources or warning message depending upon the valid/invalid users. The steps during authentication are as follows;

1) The requesting node gets public key of another node from middle server after the required credentials are valid.
2) Node A calculates signature 'S' and sends it to another Node B with whom Node A wants to communicate as;

   $A \leftarrow h(M)^{Eb} \pmod{N_b}$, where M is the message to be signed.

   $S \leftarrow A^{Da} \bmod N_a$
3) Finally Node B performs the following;

   $B \leftarrow S^{Ea} \bmod N_a$ and $Q \leftarrow B^{Db} \bmod N_b$

   Calculate h(M) , if Q = h(M), then the Node A is valid node otherwise it is treated as invalid node.

The working mechanism of this approach is depicted in the following figure as,

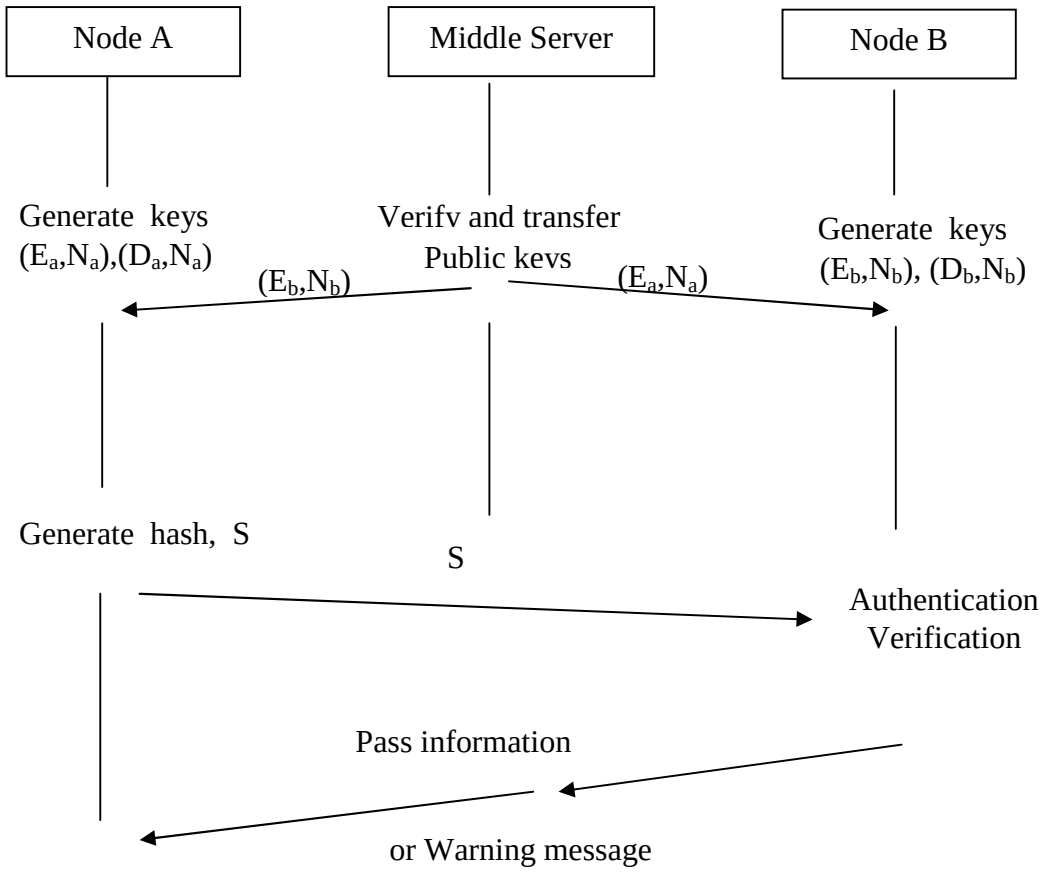

Figure 3.4: Protocol flow of RSA based indirect authentication

3.3 Kaman Authentication Mechanism

This mechanism is a type of indirect authentication system. The Kaman authentication system is a pure managed Kerberos assisted authentication and is based on the time-tested and widely deployed Kerberos protocol [2]. To support the more challenging demands of ad-hoc networks the Kaman authentication system can be useful as a secure authentication scheme. Kaman uses multiple Kerberos servers for distributed authentication and load distribution [1, 2]. The servers uses the hash of secrete keys or passwords that are only known by the users and all servers share a secret key with each other server.

When a client node C1 wants to communicate with another client node C2 then the node C1 sends request to one of the Kaman servers S. A ticket, which contains the session key for the requested communication is generated by the server and the server sends this ticket to the node C1. The client C1 in turn sends this ticket to the target client C2 with which communication is desired. Then after the client C2 acknowledges the ticket and a secure session is established between the two clients C1 and C2 using the session key provided by the server [1]. The flow of the protocol is described symbolically as follows:

1. C1 → S

Options, ID_{C1}, ID_{C2}, Times, Nonce

2. S →C1

ID_{C1}, $Ticket_{C2}$, $\{K_{C1,C2}$, Times, Nonce, $ID_{C2}\}K_{C1}$

3. C1 →C2

Options, $Ticket_{C2}$, $Authenticator_{C1}$

4. C2 → C1

$\{TS, Subkey, Seq\#\}K_{C1,C2}$, where,

$Ticket_{C2} = \{Flags, K_{C1,C2}, ID_{C1}, AD_{C1}, Times\}K_{C2}$

$Authenticator_{C1} = \{ID_{C1}, TS\}K_{C1,C2}$

ID_{C1}: Identity of Client1

ID_{C2}: Identity of Client2

AD_{C1}: Network Address of Client1

K_{Cn}: Encryption key based on hashed password of user n

$K_{C1,C2}$: Session key between Client1 and Client2

TS: Informs of time when this authenticator was generated.

The overall flow of the Kaman authentication protocol can be illustrated as;

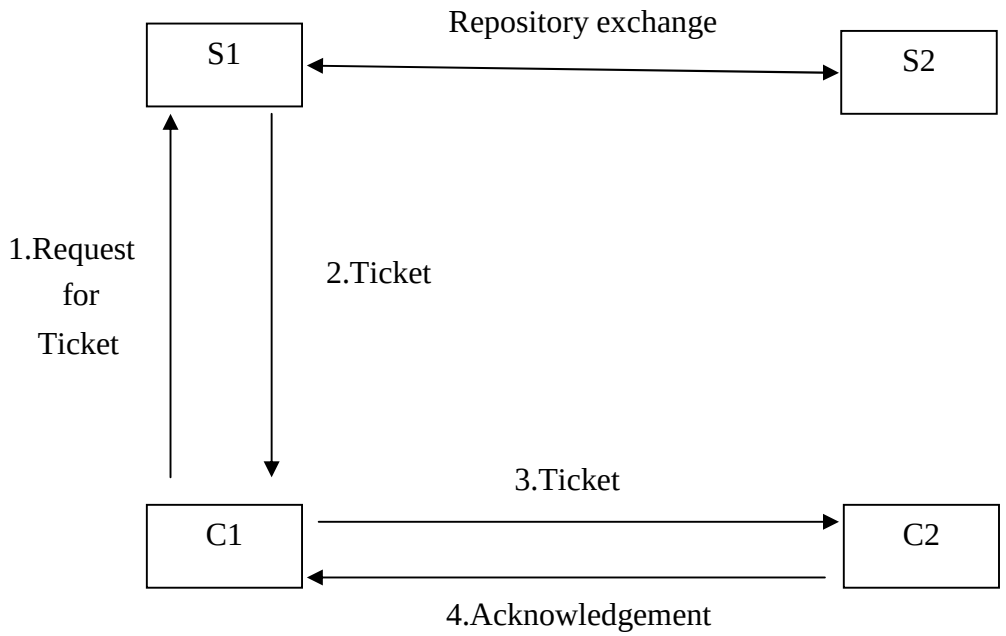

Figure 3.5: Protocol flow of Kaman authentication mechanism

During the authentication service exchange, C1 requests a ticket from the Kaman server (S1) for further communication with C2. The server first checks that if C1 and C2 have a valid lifetime associated with their user IDs. If they have valid lifetimes then the server responds by providing C1 a ticket to access C2. This ticket, along with other values, contains a session key for communication between C1 and C2. C1 then passes this ticket to C2 for establishing secure communication using the session key. C2 acknowledges receipt of this ticket by sending a Time Stamp back to C1.

3.6 Chang-Cheng's Authentication Mechanism

In this authentication approach the system consists of one trusted registration center (RC), users (U_i) and service providers (SPj). RC computes n secret keys $KRS_j = H(SID_j \| k)$ choosing a secrete key k and keys shared with each service provider SP_j , where SID_j is SP_j's identifier and $H(\cdot)$ is a private one-way hash function only known by RC. The scheme is composed of the

following phases: registration, login, authentication and key agreement, and password modification [6, 7, 8].

Registration phase: To use all resources in this system, the user must register at the RC . According to the author in [6] the steps involved in this process are as follows:

- U_i selects an identifier id_i and a password pw_i, and then sends the message {id_i, pw_i, Personal Information} to the RC through a secure channel.
- RC checks user's credentials and the personal information. If is invalid, the server rejects the registration; otherwise it is valid for next step of registration phase.
- A transformed identifier $TID_i = T_i||id_i$ as U_i's account number is then generated by RC and saves it in the database, where T_i is the registration time.
- After then RC computes $TPW_i = h(pw_i)$ and $\sigma_i = H(TID_i||k) + pw_i$ and embeds TID_i, $h(.)$, TPW_i, and σ_i in to a single block. RC subsequently sends this block to U_i.

Login phase: During login phase user U_i must enters key in the password pw_i* and following computation happens at the side of U_i

- Compute $h(pw_i^*)$ and compare this value with TPW_i. If they are same then goe to next step otherwise log in process has been stopped.
- Computes $\alpha = h(\sigma_i + pw_i^* + N_U + SID_j)$, where N_U is a nonce chosen the user U_i. {TID_i , α, N_U } is then transmitted to SP_j by the user U_i for further process.
- SP_j computes $\beta = h(KRS_j + Ns)$ and sends { TID_i , α, N_U, SID_j , β , Ns } to RC , where Ns is a nonce chosen by SP_j.

Authentication and key agreement phase: After getting the login request message, RC, SP_j and U_i execute the following procedure to complete mutual authentication and coordinate a common session key shared between SP*j* and U_i [6, 7] .

- Freshness of N_U and Ns and validity of account number TID_i is checked. If they are invalid then rejected otherwise go for next step.
- RC computes and verifies $h(H(TID_i||k) + N_U + SID_j) = \alpha$ and $h(H(SID_j||k) + Ns = \beta$. If either one does not hold, then RC terminates the connection otherwise SP*j* and U_i are considered as legal participants.

- RC subsequently chooses a random number *ran* and computes

  $\beta' = h(H(SID_j||k) + N_R)$,

  $\alpha' = h(H(TID_i||k) + N_R)$,

  $\Upsilon s = h(H(SID_j||k))+(ran|| h(H(TID_i||k)))$ and

  $\Upsilon_U = (h(H(SID_j||k))||ran) + h(H(TID_i||k))$,

  where N_R is a nonce selected by RC. Finally RC sends { β', N_R , α', Υs, Υ_U }to SP*j*.
- Freshness of N_R is checked by SP*j*. If yes, then the next step occurs otherwise procedure is stopped.
- SP*j* then computes and verifies $h(KBS_j + N_R) = \beta'$. If it holds go to next step otherwise procedure terminates.
- SP*j* computes

  $s = \Upsilon s + h(KRS_j)$ and

  $SK = h((h(KRS_j||s)+ N_U + Ns + N_R)$, where *SK* is the common session key. Finally SP*j sends* { α' , N_R, Υ_U } to U*i* .
- After getting message to U_i, the freshness of N_R is checked at U_i and then U_i computes $h(\sigma_i + pw_i^* + N_R) = \alpha'$. If they do not hold then authentication fails; otherwise SP*j* is authenticated by U_i. Then after U_i computes

  $u = \Upsilon_U + h(\sigma_i + pw_i^*)$ and

  $SK = h((u|| h(\sigma_i + pw_i^*)) + N_U + Ns + N_R)$

Password modification phase: For changing the password U_i need to enter pw_i^*. Then, U_i computes and checks $h(pw_i^*) = TPW_i$. If the condition holds then U_i can choose pw_i^{new} and calculates following values as:

$TPW_i^{new} = h(pw_i^{new})$ and

$\sigma_i^{new} = \sigma_i + pw_i^* + pw_i^{new}$

At last old values TPW_i and σ_i are replaced with new values TPW_i^{new} and σ_i^{new} respectively and U_i saves these new values for future processing.

CHAPTER 4

4. Implementation and Testing

J2ME™ (Java™ 2 Micro Edition) is a Java-based application platform developed by Sun Microsystems and is specially developed for focusing on the devices which have limited processing power and storage capabilities and intermittent or fairly low-bandwidth network connections. J2ME provides a robust, flexible environment for applications running on mobile and embedded devices. J2ME contains a subset of the APIs of Java™ Standard Edition. The targeted devices that J2ME supports include mobile phones, pagers, wireless devices and set-top boxes among others. The Mobile Information Device Profile (MIDP) and the Connected Limited Device Configuration (CLDC) define the available APIs. NetBeans 6.8 supports CLDC 1.1 and MIDP 2.1 and in this work these are used for module implementation. General cryptographic API for multi-precision computations like the BigInteger and SecureRandom classes are not supported within these so the APIs provided by bouncycastle.org have been used to support those computations [23].

4.1 Java™ 2 Micro Edition Overview

For implementation of this work, a programming language that supports platform independent code i.e. Java™ programming language is used. Mainly the Java™ technology is separated into three platforms: J2EE for server side development, J2SE for standard desktop applications and J2ME™ for resource constrained devices that are frequently used in mobile network [5]. Several different configurations and profiles with each configuration and profile having its own library APIs constitute the J2ME™. Configurations define the basic run-time environment as a set of Java™ language core libraries for a range of devices and a specific VM that runs on those devices [5, 35]. For small, resource-constrained devices such as cell phones and low-end PDAs a configuration i.e. Connected Limited Device Configuration (CLDC) is designed in J2ME™ [5]. In J2ME™ profiles are built on top of configurations, and define more advanced, device-specific API libraries. Java™ applications written for a specific profile can be ported across all the hardware/OS platforms supported by that profile. MIDP, a profile under CLDC configuration is used in this study and the applications developed in this are called MIDlets.

The Figure 4.1 shows the arrangement of J2ME™ technologies on Java™ platforms:

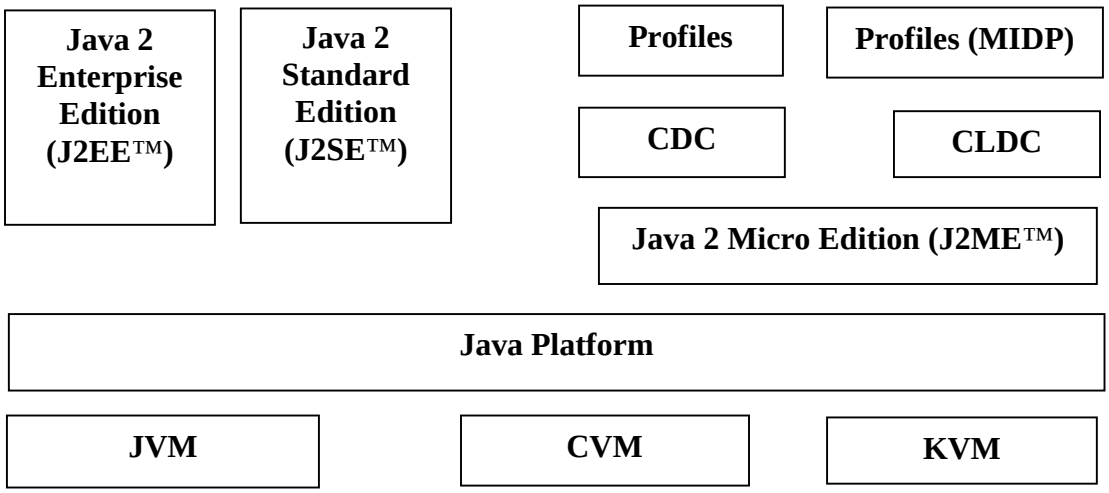

Figure 4.1: Java™ and J2ME™ Technologies [5]

4.2 Third Party Lightweight Crypto API

The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. The package is organized so that it contains a light-weight API suitable for use in any environment (including J2ME) with the additional infrastructure to conform the algorithms to the JCE Java™ Cryptography Extension) framework [5]. The BC lightweight crypto API developed by BC developers to be wrapped in BC JCE provider classes. This BC lightweight crypto API can also be used standalone, with minimum dependence on other J2SE classes [5, 23]. The downloaded Bouncy Castle J2ME™ package contains the implementation of the BC lightweight API and two core Java™ classes java.math.BigInteger and java.security.SecureRandom but these two core Java™ classes are not supported in J2ME™ / CLDC.

In this study the Bouncy Castle APIs for J2ME™ is used partially to make support for the BigInteger and SecureRandom classes that are otherwise not supported in J2ME™. To perform big integer math operations and generating cryptographically secure random numbers, the BigInteger and SecureRandom classes in Bouncy Castle APIs are used as a J2ME™ compatible versions of the original java.math.BigInteger and java.security.SecureRandom classes

respectively. Bouncy-Castle library was chosen, as it is free being an open source community and widely used in MIDP applications [23].

For this study a latest version of bouncy castle APIs used in the implementation is Release 1.48, which is approximately 9.7 MB in size (in zipped form). Since only the room for support of BigInteger and SecureRandom class is needed, the actual size of the library used is approximately 1.54 MB. The size of the library makes it necessary to reduce the size of the MIDlet by obfuscation which removes all unused classes and methods at runtime and further renames the classes not available in MIDP to avoid conflicts with the security policy of the MIDP devices [5, 21].

4.3 Development & Emulation Environment: NetBeans Overview

For implementing the required programs for this study, the NetBeans Integrated Development Environment 6.8 has chosen as a development environment. This NetBeans IDE is used for implementation because it supports the coding, testing and deployment of applications for mobile devices configuring client and server programs that support the J2ME, CLDC/MIDP in a user-friendly interface. Further NetBeans also includes an in-built emulator platform. An emulator platform simulates the execution of an application on client and server devices. An emulator also allows to understand the user experience for an application on a particular device, and to test the portability of the application across different devices. The emulator platform used in the study was Sun Java™ Wireless Toolkit 2.5.2 for CLDC which is in-built in NetBeans 6.8. Different Client and server programs have run separately and time taken during authentication by various direct and indirect authentication systems mentioned in chapter 3 has been analyzed using NetBeans IDE.

4.4 Implementation Details

During implementation of ECC based direct and indirect authentication, the ECC finite field of key sizes 192, 224, 256 and RSA with key sizes 1536, 2048, 3072 (providing equivalent in security with ECC) have selected [28, 35, 39]. The parameters a, b and generator points G(x,y) used in ECC based authentication are taken according to the recommendation of National Institute of Standards and Technology (NIST) for reasons of efficiency [27]. Various sample messages are used for input in hash function during authentication process of RSA for testing.

Various input parameters required for Kaman Authentication Mechanism and Chang-Cheng's Authentication Mechanism are selected and fed into the programs for analysis. The authentication code generation time and authentication verification time taken by various authentication schemes under this study i.e. ECC based direct authentication, ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman Authentication Mechanism and Chang-Cheng's Authentication Mechanism are captured using the standard Java™ function System.currentTimeMillis(). The implementation is performed on J2ME™ (Java TM 2 Micro Edition) environment with Windows7, 64 bits machine having 4GB RAM, with Intel CORETM i3 processor.

4.4.1 Implementation Details of ECC Based Direct Authentication

The domain parameters for elliptic curve y2 mod p = (x3 + ax + b) mod p defined over finite field GF (p) were taken from NIST standard curves having bit sizes of 192, 224 and 256 [27]. For key generation, a random private key Ks less than order of the base point is generated and the public key is computed by multiplying Ks with the base point by using the method doubleAndAddAlgo (). This method is for calculating public key takes x and y coordinates of the base point G and the integer Ks as inputs. The bits of Ks are processed one at a time, doubling the point and if the bit is 1 adding the resulting point to the base point similar to the square-and-multiply algorithm. The code snap for authentication verification is given as;

Sample code 1

```
void isAuthenticate(BigInteger avc, BigInteger r2_node, BigInteger pubx, BigInteger puby) {
    doubleAndAddAlgo(avc, basex, basey);
    BigInteger auth_x = effi_x;
    BigInteger auth_y = effi_y;

    doubleAndAddAlgo(r2_node, pubx, puby);
    BigInteger auth1x = effi_x;
    BigInteger auth1y = effi_y.negate();
```

```
    pointAddition(auth_x, auth_y, auth1x, auth1y);
    resx = add_x;
    resy = add_y;

            if ((resx.compareTo(rcx)) == 0 && (resy.compareTo(rcy) == 0)) {
       System.out.println("\nThe base node is Authenticated.");
    } else {
       System.out.println("\nThe base node is not authentic.");
    }
  }
```

Listing 4.1: Java code to perform auth verification in ECC based direct authentication

4.4.2 Implementation Details of ECC Based Indirect Authentication

In case of ECC based indirect authentication the domain parameters for elliptic curve y2 mod p = (x3 + ax + b) mod p that were taken from NIST standard curves having bit sizes of 192, 224 and 256 [27].For key generation, three random private keys Ks, Ks(A.S.) and Ks(M.S.) less than order of the base point is generated and the public key is computed by multiplying Ks, Ks(A.S.) and Ks(M.S.) with the base point by using the method doubleAndAddAlgo (). This method is for calculating public key takes x and y coordinates of the base point G and the integer Ks as inputs. The bits of Ks, Ks(A.S.) and Ks(M.S.) are processed one at a time, doubling the point and if the bit is 1 adding the resulting point to the base point similar to the square-and-multiply algorithm. The code snap for performing the authentication verification in ECC based indirect authentication scheme is listed as follows:

Sample code 2

```
void authVerification(BigInteger ACKrand, BigInteger auth_pubx, BigInteger auth_puby)
//Calculate go-ahead token AgT and it keeps secrete by server to verify user node later
    crvparm.doubleAndAddAlgo(ACKrand, auth_pubx, auth_puby);
    BigInteger temp1_Agt_x = crvparm.effi_x;
```

```
        BigInteger temp1_Agt_y = crvparm.effi_y;
        crvparm.doubleAndAddAlgo(ARint, server_pubx, server_puby);
        BigInteger temp2_Agt_x = crvparm.effi_x;
        BigInteger temp2_Agt_y = crvparm.effi_y;
        System.out.println("crvparm.effi_y " + crvparm.effi_y);
        //Point AgT is calculated
        crvparm.pointAddition(temp1_Agt_x, temp1_Agt_y, temp2_Agt_x, temp2_Agt_y_neg);
        BigInteger Agt_x = crvparm.add_x;
        BigInteger Agt_y = crvparm.add_y;
                BigInteger IRc_x = new BigInteger(infromuser.readLine());
        BigInteger IRc_y = new BigInteger(infromuser.readLine());
                //Negate the point AgT
        BigInteger Agt_y_neg = crvparm.add_y.negate();
        crvparm.pointAddition(ATc_x, ATc_y, Agt_x, Agt_y_neg);
        BigInteger final_x = crvparm.add_x;
        BigInteger final_y = crvparm.add_y;

        //Verify whether the user node is valid or not

        if(IRc_x.subtract(final_x).equals(BigInteger.ZERO)&&
        IRc_y.subtract(final_y).equals(BigInteger.ZERO)) {
           System.out.println("\n The user node is authorized one.");
        } else {
           System.out.println("\n Unauthorized user node.");
        }
}
```

Listing 4.2: Java code to perform auth verification in ECC based indirect authentication

4.4.3 Implementation Details of RSA Based Direct Authentication

For security reason RSA uses large prime numbers during generating key pair generation than ECC. In general at least 1024-bits is taken for key generation for encryption and authentication purposes to fulfill the basic security requirements [35]. For creating large prime numbers, in Java™ the BigInteger class is available, which is used especially for dealing with such big integers. The method probablePrime() present in the BigInteger class has been used to get large prime numbers. For generating cryptographically secure pseudorandom number, above method takes two arguments i.e. number of bits of prime that is to be generated and the object of a SecureRandom class.

```
SecureRandom random = new SecureRandom();   // to create object
p = BigInteger.probablePrime(1024, random);    // returns 1024-bit prime number
```

The multiply() method available in java is used to get the multiplication of two large primes, to yield the modulus N as:

```
N = p.multiply(q);
```

The GCD of two large numbers using Euclid's algorithm and multiplicative inverse using extended Euclidean algorithm are calculated by using Java's BigInteger class as:

```
e.gcd(phi_n); and
d=e.modInverse(phi_n);
```

For getting the digest value of given input message, MessageDigest class of Java is used and code for getting digest value using SHA1 is given below:

```
MessageDigest md = MessageDigest.getInstance("SHA1");
String inputmsg = "Private and public key cryptography";
md.update(inputmsg.getBytes());
byte[] output = md.digest();
```

The digest value for given input message generated in such a way is used as a signature during authentication process. The signature generated in user node is verified in server node for identifying whether the requesting node is authentic or any other invalid user [35, 39].

The snap code for the authentication code verification for RSA based direct authentication is listed below:

Sample code 3

```
void SigVarification(BigInteger Ea, BigInteger Na, BigInteger Db, BigInteger Nb, BigInteger sig_code) {
        MessageDigest md = MessageDigest.getInstance("SHA1");
        String inputmsg = "Public key cryptography";
        md.update(inputmsg.getBytes());
        byte[] output = md.digest();
                        String hofm = bytesToHex(output);
        hashOfMsgBigInt_Server = new BigInteger(new String(hofm), 16);
      BigInteger bb = sig_code.modPow(Ea, Na);
      BigInteger Q = bb.modPow(Db, Nb);
      if (Q.compareTo(hashOfMsgBigInt_Server) == 0) {
        System.out.println("\n Signature is Valid and authenticated \n");
      } else {
        System.out.println("Invalid Signature i.e. not authenticated");
      }
   }
```

Listing 4.3: Java code to perform auth verification in RSA based direct authentication

4.4.4 Implementation Details of RSA Based Indirect Authentication

In RSA based authentication the middle server shares public key to both the communicating user nodes nodeA and nodeB. The same bit length keys as selected in direct case, are taken for this approach. The parameters p, N, e, d are calculated in similar way as in RSA based direct authentication system. The digest value for given input message generated is used as a signature during authentication process. The signature generated in user node is verified in server node for authentication. The snap code for RSA based indirect authentication i.e. authentication verification code is given as:

Sample code 4

```
void SignatureVarify(BigInteger Ea, BigInteger Na, BigInteger Db, BigInteger Nb, BigInteger sig_code) {
        MessageDigest msgDig = MessageDigest.getInstance("SHA1");
        String inputmsg = "Indirect RSA authentication";
        msgDig.update(inputmsg.getBytes());
        byte[] result = msgDig.digest();
                        String hofm = bytesToHex(result);
        hashOfMsgBigInt_Server = new BigInteger(new String(hofm), 16);
      BigInteger auth_code = sig_code.modPow(Ea, Na);
      BigInteger auth_code_F = auth_code.modPow(Db, Nb);
      if (auth_code_F.compareTo(hashOfMsgBigInt_Server) == 0) {
        System.out.println("\n User node authenticated \n");
      } else {
        System.out.println("User node is not authenticated");
      }
   }
```

Listing 4.4: Java code to perform auth verification in RSA based indirect authentication

4.4.5 Implementation Details of Kaman Authentication Mechanism

Kaman authentication system is a pure managed Kerberos assisted authentication system. Suppose client1 wants to access the resources of client2. For this client1 and client2 must be mutually authenticated. Three programs representing client1, client2 and server are created in Java for accomplishing Kaman authentication system. Two large secrete keys or passwords K_{C1} and K_{C2} are generated in client1 and client2 respectively whose identification numbers are ID_{C1}, ID_{C2} [1]. These secrete keys are generated using random number generator methods of SecureRandom class of java as:

SecureRandom random = new SecureRandom(); // for ID_{C1}

K_{C1} = new BigInteger(1024, random);

SecureRandom random = new SecureRandom(); // for ID_{C2}

K_{C2} = new BigInteger(1024, random);

A large random number called NONCE that is used in authentication is also generated in requesting node client1 as:
SecureRandom random = new SecureRandom(); // for nonce
Nonce = new BigInteger(1024, random);

The hashed value of above secrete keys are calculated in client program of client nodes and transferred to the server node. For communication between client1 and client2, the authentication server sends ticket to client1 as:

ID_{C1}, $Ticket_{C2}$, {$K_{C1,C2}$, Times, Nonce, ID_{C2} }K_{C1}
The ticket $Ticket_{C2}$ transferred to the client2 program and is calculated by using the encryption function using the hashed key of client2 as:
$Ticket_{C2}$ = {Flags, $K_{C1,C2}$, ID_{C1}, AD_{C1}, Times}K_{C2}

The client2 uses the same algorithm for decrypting the ticket received from client1 and gets ID_{C1} the identity of client1 and AD_{C1} Network Address of Client1 from TicketC2. In the implementation of Kaman authentication system, DES algorithm available in Java has used for encryption and decryption of various information block and $Ticket_{C2}$.

Sample code 5

```
void verifyAuthKaman(TicketC2, AuthenticatorC1)throws NoSuchAlgorithmException {
   Cipher desCip;
   desCip = Cipher.getInstance("DES/ECB/PKCS5Padding");
        byte[] TicketC2test = desCip.doFinal(TicketC2);
         TicketC2test1 = TicketC2test.toString();
         doEncrypt(TicketC2test);
         IDc1_server = IDc1_final;
         ADc1_Server = ADc1_final;
```

```
        TimesS1 = timesServer;
        doEncrypt(AuthenticatorC1);
        IDc1 = IDc1Client;
        TS =  TSClient;
        if(IDc1_server.compareTo(IDc1)==0&& TimesS1.compareTo(TS)==0) {
                System.out.println("Authenticated");
        }
        else{
                System.out.println("Not Authenticated");
        }
}
```

Listing 4.5: Java code to perform auth verification in Kaman based authentication

4.4.6 Implementation Details of Chang-Cheng's Authentication Mechanism

In the implementation of Chang-Cheng's Authentication Mechanism, three modules represent ting users (Ui), registration center (RC) and service providers (SPj) are created using J2ME. Initially, RC chooses a private key k and computes secret keys KRSj = H(SIDj||k) shared with service provider SPj , where SIDj is SPj's identifier [6]. In the calculation of σ_i, SHA1 included in java has been used. The RC computes a transformed identifier TIDi and TPWi as:

TIDi = Ti||idi ,where Ti is the registration time for user idi and
TPWi = h(pwi) and $\sigma_i = H(TID_i||k) + pw_i$.

The SHA1 a public has function in this scheme is used for hashing the different calculations. The nonce parameters N_U , Ns, N_R are calculated using java function as:

```
SecureRandom rand = new SecureRandom();
BigInteger N_U = new BigInteger(1024, rand);
BigInteger Ns = new BigInteger(1024, rand);
BigInteger N_R = new BigInteger(1024, rand);
```

In various intermediate calculation XOR operation is needed to get the value of α, β, α ’, β’ etc .

SP*j* finally computes

$s = \Upsilon s + h(KRS_j)$ and

$SK = h((h(KRS_j \| s) + N_U + Ns + N_R)$, where *SK* is the common session key.

Finally SP*j sends* { α ’, N_R, Υ_U } to U*i* .

At last the User U_i computes $h(\sigma_{i+} pw_i * + N_R) = \alpha$ ’. If it is valid then authentication process completes and the user U_i uses the resources of service providers (SPj).

Sample code 6

```
void verifyAuthetication(pwd, TId,)throws NoSuchAlgorithmException {
   SecureRandom rand = new SecureRandom();
   MessageDigest h = MessageDigest.getInstance("SHA1");

    // Digest the message using SHA1 instance of hash function
   h.update(pwd.getBytes());
   byte[] output = h.digest();
   String TPW = String.valueOf(output);

   //Concatenating TId with k i.e. Tid||k
   BigInteger k = new BigInteger(2048, rand);
   String temp1 = TId.concat(k.toString());
   BigInteger temp1Big = new BigInteger(temp1);

   //Digest the TId||k with another hash function
   MessageDigest H = MessageDigest.getInstance("SHA256");
   H.update(temp1Big.toString().getBytes());
   byte[] outputTemp = H.digest();
   String testTemp = String.valueOf(outputTemp);
   BigInteger testTempBig = new BigInteger(testTemp);
   BigInteger sigma = testTempBig.xor( new BigInteger(pwd));
```

```
String alphaStr  = inFromSP.readLine();
BigInteger alpha = new BigInteger(alphaStr);
String betaStr  = inFromSP.readLine();
BigInteger beta = new BigInteger(betaStr);
String NuStr  = inFromSP.readLine();
BigInteger Nu = new BigInteger(NuStr);
String NsStr  = inFromSP.readLine();
BigInteger Ns = new BigInteger(NsStr);
BigInteger AuthenticateTemp1 = sigma.xor(Nu).xor(SId);
BigInteger AuthenticateTemp2 = KRSBig.xor(Ns);
//Compare AuthenticateTemp1 and AuthenticateTemp2 with alpha and beta respectively.
if(AuthenticateTemp1.compareTo(alpha)==0&&
AuthenticateTemp2.compareTo(beta)==0) {
        System.out.println("Authenticated");
}
else{
        System.out.println("Not Authenticated");
}
}
```

Listing 4.6: Java code to perform auth verification in Chang-Cheng's Auth Mechanism

4.5 Sample Test Cases

In the testing, the input data is fed into the authentication code generation and authentication verification modules and the result for valid or invalid authentication is obtained. The authentication code generation time and authentication verification time taken by various authentication schemes under this study are captured using the standard Java™ function System.currentTimeMillis(). The sample data for authentication code generation and authentication verification system in various authentication schemes under study are listed below:

1) ECC Based Direct Authentication

Authentication code generated:

486681023306145253851861224975664750760200112688668906014 9,

251028355464893650109249842546676730272723696533331243517 5

Authentication Verification code generated:

486681023306145253851861224975664750760200112688668906014 9,

251028355464893650109249842546676730272723696533331243517 5

Authentication Code Generation took 218 milliseconds.

Authentication verification took 201 milliseconds.

2) ECC Based Indirect Authentication

Authentication code generated:

187034996565008388562143038756185615909405338966153425092704621046652519067 74,

820419884545004225622973060731774191994125921766019085638546713566751718640 06

Authentication Verification code generated:

187034996565008388562143038756185615909405338966153425092704621046652519067 74,

820419884545004225622973060731774191994125921766019085638546713566751718640 06

Authentication Code Generation took 235 milliseconds.

Authentication verification took 213 milliseconds.

3) RSA Based Direct Authentication

Authentication code generated:

601582705447551055419695365108830471249945045 45

Authentication Verification code generated:

601582705447551055419695365108830471249945045 45

Authentication Code Generation took 287 milliseconds.

Authentication verification took 264 milliseconds.

4) RSA Based Indirect Authentication

Authentication code generated

80155958522242270269896167685540754423569189544667 43

Authentication Verification code generated:

80155958522242270269896167685540754423569189544667 43

Authentication Code Generation took 293 milliseconds.

Authentication verification took 271 milliseconds.

5) Kaman Authentication Mechanism

Authentication code generated:

55546008943817740293915197451784769108058161 19

Authentication Verification code generated:

55546008943817740293915197451784769108058161 19

Authentication Code Generation took 186 milliseconds.

Authentication verification took 165 milliseconds.

6) Chang-Cheng's Authentication Mechanism

Authentication code generated:

503322936220314048575522840485755228021941036 4023

Authentication Verification code generated:

503322936220314048575522840485755228021941036 4023

Authentication Code Generation took 203 milliseconds.

Authentication verification took 183 milliseconds.

CHAPTER 5

5. Analysis

The analysis of various direct and indirect authentication schemes under this study i.e. ECC based direct authentication, ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman Authentication Mechanism and Chang-Cheng's Authentication Mechanism, is done by analyzing time taken during the authentication code generation and authentication verification using the standard Java™ function System.currentTimeMillis(). The computational cost analysis of specified authentication schemes is done by varying the Key sizes and average time is taken by running the modules 20 times and taken a rounded average time so that the result and analysis becomes more realistic and accurate. In case of ECC based authentication the NIST recommended domain parameters are used during public key generation and other intermediate calculations during authentication process. In the analysis of RSA based authentication, the key sizes equivalent to ECC with similar security level are used. For Kaman authentication and Chang Cheng's authentication schemes, the secrete keys and nonce values are taken as equivalent key sizes to RSA that are used in various intermediate operations during authentication.

5.1 Empirical Analysis

The average time is taken from 20 experiments that are performed for capturing the time taken during authentication code generation and authentication code verification of all authentication schemes under study. The time taken (in milliseconds) by different authentication schemes i.e. ECC based direct authentication, ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman authentication mechanism and Chang Cheng's authentication mechanism for comparative analysis are listed in the following tables and corresponding graphs built from these tables are also given in the following sections.

5.1.1 Authentication Code Generation and Authentication Verification Time Analysis for Key size in bits 192 / 1536

Algorithm Used	Auth Code Generation Time	Auth Verification Time
ECC Direct Auth	218	201
ECC Indirect Auth	235	213
RSA Direct Auth	287	264
RSA Indirect Auth	293	271
Kaman Auth	186	165
Chang-Cheng Auth	203	183

Table 5.1: Authentication Code Generation and Authentication Verification Time (in milliseconds) for key size in bits 192 / 1536

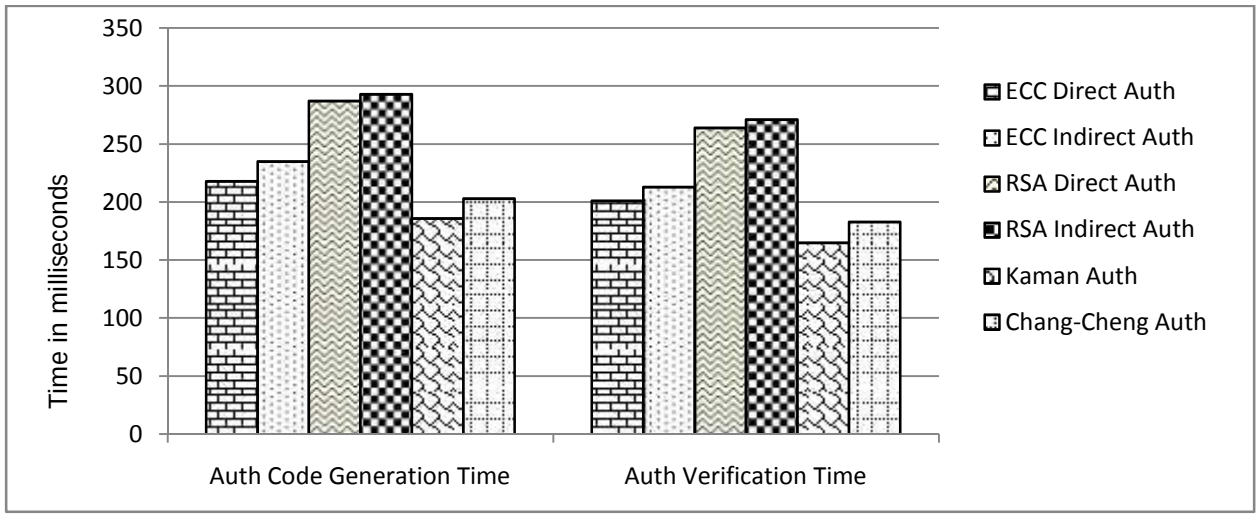

Figure 5.1: Graph shows authentication code generation and authentication verification time for key size in bits 192 / 1536 for specified algorithms.

From the above graph it can be concluded that the Kaman authentication scheme takes the minimum time in comparison to other authentication schemes during authentication code generation. The authentication code generation time seems in increasing order for Chang-Cheng's authentication, ECC based direct authentication, ECC based indirect authentication and

RSA based direct authentication. The RSA based indirect authentication takes highest time than other algorithms in authentication code generation as well as in authentication verification. The time taken during authentication verification is less than the time taken in authentication code generation for all authentication schemes mentioned in Table 5.1. RSA based indirect authentication takes slightly higher time than RSA based direct authentication but there is large variance in time in comparison to Kaman authentication mechanism.

5.1.2 Authentication Code Generation and Authentication Verification Time Analysis for Key size in bits 224 / 2048

Algorithm Used	Auth Code Generation Time	Auth Verification Time
ECC Direct Auth	247	224
ECC Indirect Auth	291	264
RSA Direct Auth	453	389
RSA Indirect Auth	472	406
Kaman Auth	259	230
Chang-Cheng Auth	265	239

Table 5.2: Authentication Code Generation and Authentication Verification Time (in milliseconds) for key size in bits 224 / 2048

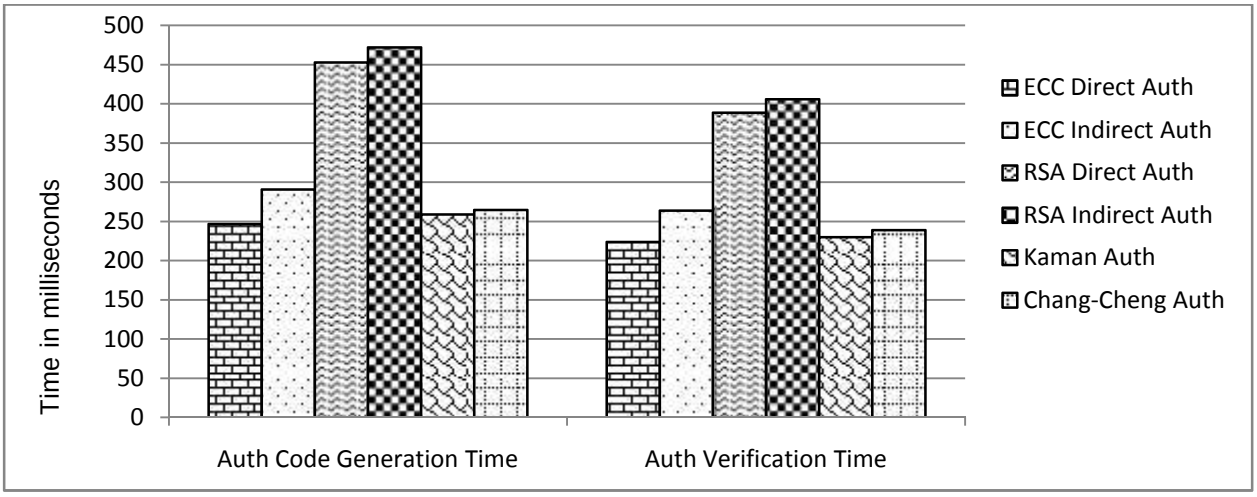

Figure 5.2: Graph shows authentication code generation and authentication verification time for key size in bits 224 / 2048 for specified algorithms

From the graph Figure 5.2, after increasing key size from 192 / 1536 to 224 / 2048, from above graph there seems not so much drastic change in the time taken in all other algorithms except RSA based direct and RSA based indirect authentication where significant time increase with key size increase. For other systems time taken for authentication code generation and authentication verification has increased smoothly with the increase in key size.

5.1.3 Authentication Code Generation and Authentication Verification Time Analysis for Key size in bits 256 / 3072

Algorithm Used	Auth Code Generation Time	Auth Verification Time
ECC Direct Auth	369	317
ECC Indirect Auth	448	385
RSA Direct Auth	674	528
RSA Indirect Auth	697	559
Kaman Auth	375	325
Chang-Cheng Auth	406	357

Table 5.3: Authentication Code Generation and Authentication Verification Time (in milliseconds) for key size in bits 256 / 3072

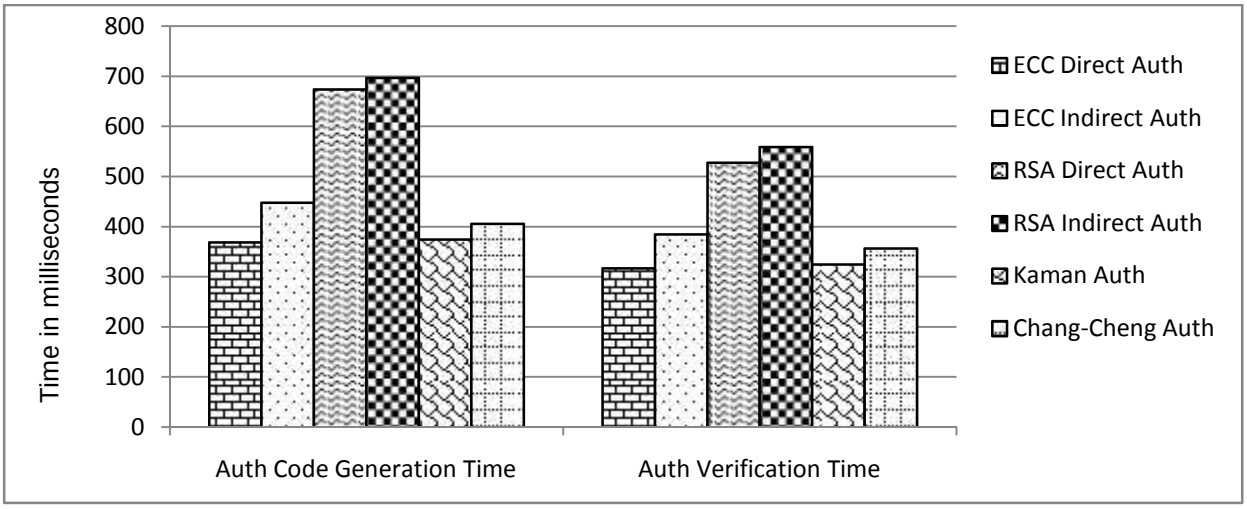

Figure 5.3: Graph shows authentication code generation and authentication verification time for key size in bits 256 / 3072 for specified algorithms

From above graph, it can be seen that there is smooth increment in time taken for authentication code generation and authentication verification for Kaman authentication mechanism, Chang Cheng's authentication mechanism and ECC based direct authentication. Also, from graph, both the authentication code generation and authentication verification time for ECC based indirect authentication increases drastically in comparison to ECC based direct authentication when considering key sizes 192 / 1536 and 256 / 3072. From above graph it is clear that the Kaman and Chang Cheng's authentication schemes takes higher time than ECC based direct authentication and at this point RSA based indirect authentication takes maximum time in both authentication code generation and authentication verification time.

5.2 Final Result

ECC based direct authentication system seems clearly an efficient authentication system than other authentication systems i.e. ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman Authentication Mechanism and Chang-Cheng's Authentication Mechanism analyzed under this study for mobile networks. In case of ECC based direct and indirect authentication systems, when the key size is increased the time taken during authentication code generation and authentication verification increases linearly where as in RSA based direct and indirect authentication schemes a drastic increase in time for authentication code generation and authentication verification has occurred. In overall authentication process including authentication code generation and authentication verification, ECC based authentication system seems superior over RSA based authentication. When key size is small i.e. key size in bits 192 / 1536, the Kaman and Chang Cheng's authentication system takes minimum time during authentication code generation and authentication verification than ECC based direct and indirect authentication systems and RSA based direct and indirect authentication systems. When the key size in bits is increased i.e. 224 / 2048 and 256 / 3072, the time taken for authentication code generation and authentication verification by Kaman and Chang Cheng's authentication systems is higher than the ECC based direct authentication and smaller than the ECC based indirect and RSA based direct and indirect authentication systems. At this point ECC based direct authentication takes minimum time for overall authentication process than all other authentication systems under this study. Finally, after the computational cost analysis of various direct and indirect authentication schemes in mobile network, the ECC

based direct authentication system is found as an efficient authentication system than other ECC based indirect authentication, RSA based direct authentication, RSA based indirect authentication, Kaman Authentication Mechanism and Chang-Cheng's Authentication Mechanism analyzed under this study.

CHAPTER 6

6. Conclusion and Future Work

The conclusion can be derived directly from the final result of implementation. However some more supporting facts of ECC based direct authentication systems have been included below. The utility of this study and future works are included in the recommendation section.

6.1 Conclusions

The various direct and indirect authentication systems i.e. ECC based direct authentication system, ECC based indirect authentication system, RSA based direct authentication system, RSA based indirect authentication system, Kaman authentication mechanism and Chang Cheng's authentication systems were implemented on J2ME™ platform. The computational cost of authentication code generation and authentication verification were analyzed separately for the above direct and indirect authentication schemes. After analysis it can be concluded that the ECC Based Direct Authentication System seems clearly an efficient authentication system in case of mobile networks than other authentication systems analyzed under this study. Also, for overall authentication process including authentication code generation and authentication verification, the ECC based authentication system seems superior to other authentication systems under this study by considering computational cost and the security of the system. The ECC based direct authentication system had taken the advantage of elliptic curve properties and hence it provides the secure environment in mobile networks.

6.2 Recommendation

In case of mobile networks, the ECC based direct authentication system has seen as an efficient and secure authentication system after completion of this study. The recommendations after this study are:

- The implementation of authentication systems analyzed under this study is done on J2ME™ using a third-party Bouncy-Castle's API. One can build up their own.
- This study can be used for further study for better performance of other token and image based authentication systems.

- For ECC authentication approaches, this study can be extended for other standard curves like NIST recommended curves.
- This study can be used for further study of Biometric authentication systems and smart card based authentication systems.

References

[1] A. A. Pirzada and C. McDonald, “Kerberos Assisted Authentication in Mobile Ad-hoc Networks”, 27th Australasian Computer Science Conference, Vol. 26, pp:41-46, 2004.

[2] A. Fox, and S.D. Gribble, “Security on the move: Indirect Authentication using Kerberos”, In proceeding of the Second Annual International Conference on Mobile Computing and Networking, pp: 155-164, 1996.

[3] A. Hämäläinen, P. Jäppinen, J. Porras, "Applying Wireless Technology to an Access control system", White papers, Lappeenranta University of Technology, September 2003.

[4] A. Khalique, K. Singh and S. Sood,“A Password-Authenticated Key Agreement Scheme Based on ECC Using Smart Cards”, IJCA, Volume 2 – No.3, May 2010.

[5] B. Kayayurt, “End-to-End Security for Mobile Devices”, Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey, July 2004.

[6] C. C. Chang and T. F. Cheng, “A robust and efficient smart card based remote login mechanism for multi-server architecture”, International Journal of Innovative Computing, Information and Control, vol. 7, pp. 4589-4602, 2011.

[7] C. T. Li, “A new password authentication and user anonymity scheme based on elliptic curve cryptography and smart card”, IET Information Security, 2012.

[8] C. T. Li, C. C. Lee and C. W. Lee, “An improved two-factor user authentication protocol for wireless sensor networks using elliptic curve cryptography”, Sensor Letters, 2012.

[9] C. T. Li, C. C. Lee, H. Mei and C. H. Yang, “A Password and Smart Card Based User Authentication Mechanism for Multi-Server Environments” International Journal of Future Generation Communication and Networking Vol. 5, No. 4, December, 2012

[10] C. T. Li, C. Y. Weng and C. I. Fan, “Two-factor user authentication in multi-server networks”, International Journal of Security and Its Applications, vol. 6, pp. 261-267, 2012.

[11] Certicom Research, “Standards for Efficient Cryptography SEC1: Elliptic Curve Cryptography”, Version 1.0, September 20, 2000.

[12] Certicom Research, “Standards for Efficient Cryptography SEC2: Recommended Elliptic Curve Domain Parameters”, Version 1.0, September 20, 2000.

[13] D. Wang, C. Ma , “Cryptanalysis of a remote user authentication scheme for mobile client–server environment based on ECC”, Information Fusion, pp 498-503, , October 2013.

[14] H.C.A.V. Tilborg, *Fundamentals of Cryptology*, Kluwer Academic Publisher Boston, 1988.

[15] H. Kyusuk, K. Kwangjo and S. Taeshik, "Untraceable Mobile Node Authentication in WSN", *Journal of Open Access sensors*, Vol. 10, pp.4410-4429, 2010.

[16] J. A. Saraireh and S. Yousef, "Authentication Transmission Overhead Between Entities in Mobile Networks", *In Proceedings of International Journal of Computer Science and Network Security*, vol. 6, no. 3B, March 2006.

[17] J. H. Lee and D. H. Lee, Efficient and secure remote authenticated key agreement scheme for multi-server using mobile equipment, Proc. of the 26th International Conference on Consumer Electronics,Las Vegas, USA, pp.1-2, 2008.

[18] J. Muthukuru, B. Sathyanarayana “A Survey of Elliptic Curve Cryptography Implementation Approaches for Efficient Smart Card Processing”, Global Journal of Computer Science and Technology, Volume 12- Version 1.0, January 2012.

[19] J. Portilla, A. Otero, E. de la Torre, T. Riesgo, O. Stecklina, S. Peter and P. Langendorfer, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors", *International Journal of Distributed Sensor Networks*, Vol. 2010, pp. 1-12, 2010.

[20] K. K. Goyal and M.S. Chahar, "A Novel Remote User Authentication Scheme using Smart Card with Biometric Based on ECDLP" , International Journal of Information Technology and Knowledge Management, Volume 4, No.2, pp. 649-651, July-December 2011.

[21] L. P. Pandey and J. Bhatta, "Performance Evaluation of RSA Variants and Elliptic Curve Cryptography", International Journal of Computer Science and Network Security (IJCSNS), Vol.11, No.11, 2011.

[22] L. Yang, J. Han, Y. Qi and Y. Liu, "Identification-Free Batch Authentication for RFID Tags", *In Proceedings of the IEEE International Conference on Network Protocols*, 2010.

[23] Legion of the Bouncy Castle, *The Bouncy Castle Crypto APIs for Java,* www.bouncycastle.org, 2011.

[24] M. Masoumi and H. Mahdizadeh, *A Novel and Efficient Hardware Implementation of Scalar Point Multiplier*, Iranian Journal of Electrical & Electronic Engineering (IJEEE), Vol. 8, No. 4, 2012.

[25] N. Boertien, E.M. Middelkoop, "Authentication in mobile applications", *Enschede: Telematica Institute*, 2001.

[26] N. Vijayarangan, "A System and Design of Extensible Authentication Protocols Based on ECC and SKE Mechanisms for Mobile and Wireless Communications", *In Proceedings of*

the 9th WSEAS International Conference on Advances in E-Activities, Information Security and Privacy, pp. 53-57, 2010.

[27] National Institute of Standard and Technology, *Recommended Elliptic Curves for Federal Government Use,* 1999.

[28] P. G.Rajeswari and K. Thilagavathi, "An Efficient Authentication Protocol Based on Elliptic Curve Cryptography for Mobile Networks", *International Journal of Computer Science and Network Security,* Vol. 9, No.2, pp. 176-185, 2009.

[29] P. G.Rajeswari and K. Thilagavathi, A Novel Protocol for Indirect Authentication in Mobile Networks based on Elliptic Curve Cryptography", *Journal of Theoretical and Applied Information Technology,* Vol.6, No.1, p.p. 56 – 66, 2009.

[30] R. Anitha, S. Ravimaran, P. Valarmathi, *Security Model that Prevents Data Leakage in Distributed Mobile Systems Using Surrogate Objects,* Special Issue of International Journal of Computer Applications (0975 – 8887) on Advanced Computing and Communication Technologies for HPC Applications - ACCTHPCA, 2012.

[31] R. A. Mollin, "An Introduction to Cryptography", Second Edition, Chapman & Hall/CRC, Taylor & Francis Group, 2007.

[32] R. Abdellatif, H. K. Aslan and S. H. Elramly, "New Real Time Multicast Authentication Protocol", *International Journal of Network Security*, Vol.12, No.1, pp.13-20, Jan 2011.

[33] R. C. C. Cheung, N. J. Telle, W. Luk and P. Y. K. Cheung, "Customizable Elliptic Curve Cryptosystems", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 13, No. 9, pp. 1048-1059, 2005.

[34] R.C.C. Cheung, W. Luk and P. Y.K. Cheung, "Reconfigurable Elliptic Curve Cryptosystems on a Chip", *In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Vol. 1, pp. 24-29, 2005.

[35] R. Soram and M. Khomdram, “Juxtaposition of RSA and Elliptic Curve Cryptosystem”, International Journal of Computer Science and Network Security, Vol. 9 No.9, September 2009.

[36] T. Struk, “Elliptic Curve Cryptography as Suitable Solution for Mobile Devices”, National University of Ireland, Galway, August 28, 2009.

[37] V. Vijayalakshmi and Dr. T.G. Palanivelu ,"Secure Localization Using Elliptic Curve Cryptography in Wireless Sensor Networks", IJCSNS International Journal of Computer Science and Network Security, Vol.8 No.6, June 2008.

[38] W. Chou, “Elliptic Curve Cryptography and Its Applications to Mobile Devices”, University of Maryland, Department of Mathematics, College Park.

[39] W. Stallings, “Cryptography and Network Security Principles and Practices” Fourth Edition. 2009.

Appendix
(Code for Implementation)

(1) ECC Based Direct Authentication

Key Generation

```
void keyGeneration() {
        SecureRandom random = new SecureRandom();
        ks = new BigInteger(pbitlen, random);
        //Public key generatiion
        doubleAndAddAlgo(ks, basex, basey);
        pubx = effi_x;
        puby = effi_y;
    }
```

Point Doubling and Point Addition

```
void doubleAndAddAlgo(BigInteger a1, BigInteger x, BigInteger y) {
        BigInteger ks1 = a1;
        int bitlen = a1.bitLength();
        int count = bitlen;
        int count1 = bitlen;
        BigInteger[] bits = new BigInteger[count];
        while (count > 0) {
            count--;
            BigInteger u = ks1.mod(two);
            bits[count] = u;
            ks1 = ks1.divide(two);
        }
        effi_x = x;
        effi_y = y;
        for (int j = 1; j < count1; j++) {
            pointDoubling(effi_x, effi_y);
            effi_x = mulx;
            effi_y = muly;
            if ((bits[j].compareTo(one)) == 0) {
                pointAddition(x, y, effi_x, effi_y);
```

```
            effi_x = add_x;
            effi_y = add_y;
        }
    }

}

void pointDoubling(BigInteger x1, BigInteger y1) {
        u1 = ((three.multiply(x1).multiply(x1)).add(a)); //3*x*x+a
        v1 = ((two.multiply(y1)).modInverse(p));//2*y
        slope = (u1.multiply(v1)).mod(p);
        mulx = ((slope.multiply(slope)).subtract(two.multiply(x1))).mod(p);
        muly = ((slope.multiply(x1.subtract(mulx))).subtract(y1)).mod(p);
}
void pointAddition(BigInteger x1, BigInteger y1, BigInteger x2, BigInteger y2) {
        u1 = (y2.subtract(y1));
        v1 = (x2.subtract(x1)).modInverse(p);
        slope = (u1.multiply(v1)).mod(p);
        add_x = ((slope.multiply(slope)).subtract(x1).subtract(x2)).mod(p);
        add_y = ((slope.multiply(x1.subtract(add_x))).subtract(y1)).mod(p);
}
```

Requesting Code Generation

```
void rcGeneration() {
    r1_basenode = new BigInteger(pbitlen, random);   number generator for basenode
    doubleAndAddAlgo(r1_basenode, basex, basey);
    rc_x = effi_x;
    rc_y = effi_y;
}
```

Authentication verification

```
void isAuthenticate(BigInteger avc, BigInteger r2_node, BigInteger pubx, BigInteger puby) {
        doubleAndAddAlgo(avc, basex, basey);
        BigInteger auth_x = effi_x;
        BigInteger auth_y = effi_y;
        doubleAndAddAlgo(r2_node, pubx, puby);
        BigInteger auth1x = effi_x;
        BigInteger auth1y = effi_y.negate();
        pointAddition(auth_x, auth_y, auth1x, auth1y);
        resx = add_x;
        resy = add_y;
           if ((resx.compareTo(rcx)) == 0 && (resy.compareTo(rcy) == 0)) {
           System.out.println("\nThe base node is Authenticated.");
        } else {
           System.out.println("\nThe base node is not authentic.");
        }
    }
```

(2) ECC Based Indirect Authentication

Public Key Generation

```
    void publicKeyGen(BigInteger ks) {
        doubleAndAddAlgo(ks, basex, basey);
        pubx = effi_x;
        puby = effi_y;
    }
```

Point Doubling and Point Addition

```
void doubleAndAddAlgo(BigInteger a1, BigInteger x, BigInteger y) {
        BigInteger ks1 = a1;
        int bitlen = a1.bitLength();
        int count = bitlen;
```

```
    int count1 = bitlen;
    BigInteger[] bits = new BigInteger[count];
    while (count > 0) {
        count--;
        BigInteger u = ks1.mod(two);
        bits[count] = u;
        ks1 = ks1.divide(two);

    }
    effi_x = x;
    effi_y = y;
    for (int j = 1; j < count1; j++) {
    pointDoubling(effi_x, effi_y);
        effi_x = mulx;
        effi_y = muly;
        if ((bits[j].compareTo(one)) == 0) {
            pointAddition(x, y, effi_x, effi_y);
            effi_x = add_x;
            effi_y = add_y;
        }
    }
}

void pointDoubling(BigInteger x1, BigInteger y1) {
    u1 = ((three.multiply(x1).multiply(x1)).add(a));
    v1 = ((two.multiply(y1)).modInverse(p));//2*y
    slope = (u1.multiply(v1)).mod(p);
    mulx = ((slope.multiply(slope)).subtract(two.multiply(x1))).mod(p);
    muly = ((slope.multiply(x1.subtract(mulx))).subtract(y1)).mod(p);
  }
void pointAddition(BigInteger x1, BigInteger y1, BigInteger x2, BigInteger y2) {
```

```
        u1 = (y2.subtract(y1));
        v1 = (x2.subtract(x1)).modInverse(p);
        slope = (u1.multiply(v1)).mod(p);
        add_x = ((slope.multiply(slope)).subtract(x1).subtract(x2)).mod(p);
        add_y = ((slope.multiply(x1.subtract(add_x))).subtract(y1)).mod(p);
    }
```

Authentication verification

```
void isAuthenticate(BigInteger IRc_x, BigInteger IRc_y, BigInteger ATc_x, BigInteger ATc_y,
BigInteger Agt_x, BigInteger Agt_y_neg ) {
    pointAddition(ATc_x, ATc_y, Agt_x, Agt_y_neg);
    BigInteger final_x = crvparm.add_x;
    BigInteger final_y = crvparm.add_y;
if (IRc_x.subtract(final_x).equals(BigInteger.ZERO) &&
IRc_y.subtract(final_y).equals(BigInteger.ZERO)) {
            System.out.println("\n The user node is authorized one.");
        } else {
            System.out.println("\n Unauthorized user node.");
        }
    }
```

(3) RSA Based Direct Authentication

Key Generation

```
void keygen()
{
pa = BigInteger.probablePrime(bitlength / 2, rand);
qa = BigInteger.probablePrime((bitlength - (bitlength / 2)), rand);
Na = pb.multiply(qa);
phi_na = (pa.subtract(BigInteger.ONE)).multiply(qa.subtract(BigInteger.ONE));
Ea = BigInteger.probablePrime(bitlength / 2, rand);
while (Ea.gcd(phi_na).compareTo(BigInteger.ONE) != 0 && Ea.compareTo(phi_na) < 0) {
            Ea = BigInteger.probablePrime(bitlength / 2, rand);
```

```
}
//computing public key
Da = Ea.modInverse(phi_na);
}
```

Signature Generation

```
void sigGen(BigInteger Eb, BigInteger Nb) {
    try {
        NodeA objrsa = new NodeA();
        MessageDigest md = MessageDigest.getInstance("SHA1");
        String inputmsg = "Public key cryptography";
        md.update(inputmsg.getBytes());
        byte[] output = md.digest();
        String hofm = bytesToHex(output);
        hashOfMsgBigInt_Client = new BigInteger(new String(hofm), 16);
        BigInteger aa = hashOfMsgBigInt_Client.modPow(Eb, Nb);
        sig = aa.modPow(objrsa.Da, objrsa.Na);

    } catch (Exception e) {
        System.out.println("Exception: " + e);
    }
  }
```

Authentication Verification

```
void SigVarification(BigInteger Ea, BigInteger Na, BigInteger Db, BigInteger Nb, BigInteger
sig_code) {
    try {
        MessageDigest md = MessageDigest.getInstance("SHA1");
        String inputmsg = "Public key cryptography";
        md.update(inputmsg.getBytes());
        byte[] output = md.digest();
```

```
                    String hofm = bytesToHex(output);
        hashOfMsgBigInt_Server = new BigInteger(new String(hofm), 16);
        } catch (Exception e) {
        System.out.println("Exception: " + e);
    }
    BigInteger bb = sig_code.modPow(Ea, Na);
    BigInteger Q = bb.modPow(Db, Nb);
    if (Q.compareTo(hashOfMsgBigInt_Server) == 0) {
        System.out.println("\n Signature is Valid and authenticated \n");
    } else {
        System.out.println("Invalid Signature i.e. not authenticated");
    }
  }
```

(4) RSA Based Indirect Authentication

Key Generation

```
 void keyGeneration()
{       pa = BigInteger.probablePrime(bitlength / 2, rand);
        pb = BigInteger.probablePrime(bitlength / 2, rand);
        qa = BigInteger.probablePrime((bitlength - (bitlength / 2)), rand);
        qb = BigInteger.probablePrime((bitlength - (bitlength / 2)), rand);
        Na = pa.multiply(qa);
        Nb = pb.multiply(qb);
        phi_na = (pa.subtract(BigInteger.ONE)).multiply(qa.subtract(BigInteger.ONE));
        phi_nb = (pb.subtract(BigInteger.ONE)).multiply(qb.subtract(BigInteger.ONE));
        Ea = BigInteger.probablePrime(bitlength / 2, rand);
        while (Ea.gcd(phi_na).compareTo(BigInteger.ONE) != 0 && Ea.compareTo(phi_na) <
0) {
                Ea = BigInteger.probablePrime(bitlength / 2, rand);
        }
        Eb = BigInteger.probablePrime(bitlength / 2, rand);
```

```
        while (Eb.gcd(phi_nb).compareTo(BigInteger.ONE) != 0 && Eb.compareTo(phi_nb) <
0) {
                Eb = BigInteger.probablePrime(bitlength / 2, rand);
        }
        Da = Ea.modInverse(phi_na);
        Db = Eb.modInverse(phi_nb);
}
```

Signature Generation

```
void sigGen(BigInteger Eb, BigInteger Nb, BigInteger Da, BigInteger Na) {
        NodeA objrsa = new NodeA();
        MessageDigest md = MessageDigest.getInstance("SHA1");
        String inputmsg = "Modern electronic distribution networks";
        md.update(inputmsg.getBytes());
        byte[] output = md.digest();
        String hofm = bytesToHex(output);
        BigInteger hashOfMsgBigInt_Client = new BigInteger(new String(hofm), 16);
        //----  Signature Generation  ----------
        BigInteger aa = hashOfMsgBigInt_Client.modPow(Eb, Nb);
        sig = aa.modPow(Da, Na);
   }
```

Signature Verification and Authentication

```
        void SigVarification(BigInteger Ea, BigInteger Na, BigInteger Db, BigInteger Nb,
BigInteger sig_code) {
                MessageDigest md = MessageDigest.getInstance("SHA1");
                String inputmsg = "Modern electronic distribution networks";
                md.update(inputmsg.getBytes());
                byte[] output = md.digest();
                System.out.println("SHA1(\"" + inputmsg + "\") =");
                String hofm = bytesToHex(output);
```

```
            hashOfMsgBigInt_Server = new BigInteger(new String(hofm), 16);
            System.out.println("   " + hofm);
            BigInteger bb = sig_code.modPow(Ea, Na);
            BigInteger Q = bb.modPow(Db, Nb);
            if (Q.compareTo(hashOfMsgBigInt_Server) == 0) {
                  System.out.println("\n Signature is Valid so authenticated\n");
            } else {
                  System.out.println("Invalid Signature, not authenticated");
            }
   }
```

(5) Kaman Authentication Mechanism

Authentication Verification

```
void verifyAuthKaman(TicketC2, AuthenticatorC1)throws NoSuchAlgorithmException {
   Cipher desCip;
   desCip = Cipher.getInstance("DES/ECB/PKCS5Padding");
        byte[] TicketC2test = desCip.doFinal(TicketC2);
         TicketC2test1 = TicketC2test.toString();
         doEncrypt(TicketC2test);
         IDc1_server = IDc1_final;
         ADc1_Server = ADc1_final;
         TimesS1 = timesServer;
         doEncrypt(AuthenticatorC1);
         IDc1 = IDc1Client;
         TS =  TSClient;
        if(IDc1_server.compareTo(IDc1)==0&& TimesS1.compareTo(TS)==0) {
                  System.out.println("Authenticated");
        }
        else{
                  System.out.println("Not Authenticated");
        }
```

}

Listing 4.5: Java code to perform auth verification in Kaman based authentication

(6) Chang-Cheng's Authentication Mechanism

Authentication Verification

```
void verifyAuthetication(pwd, TId,)throws NoSuchAlgorithmException {
        SecureRandom rand = new SecureRandom();
        MessageDigest h = MessageDigest.getInstance("SHA1");
         // Digest the message using SHA1 instance of hash function
        h.update(pwd.getBytes());
        byte[] output = h.digest();
        String TPW = String.valueOf(output);
        //Concatenating TId with k i.e. Tid||k
        BigInteger k = new BigInteger(2048, rand);
        String temp1 = TId.concat(k.toString());
        BigInteger temp1Big = new BigInteger(temp1);
        //Digest the TId||k with another hash function
        MessageDigest H = MessageDigest.getInstance("SHA256");
        H.update(temp1Big.toString().getBytes());
        byte[] outputTemp = H.digest();
        String testTemp = String.valueOf(outputTemp);
        BigInteger testTempBig = new BigInteger(testTemp);
        BigInteger sigma = testTempBig.xor( new BigInteger(pwd));
        String alphaStr  = inFromSP.readLine();
        BigInteger alpha = new BigInteger(alphaStr);
        String betaStr  = inFromSP.readLine();
        BigInteger beta = new BigInteger(betaStr);
        String NuStr  = inFromSP.readLine();
        BigInteger Nu = new BigInteger(NuStr);
        String NsStr  = inFromSP.readLine();
        BigInteger Ns = new BigInteger(NsStr);
```

```
        BigInteger AuthenticateTemp1 = sigma.xor(Nu).xor(SId);
        BigInteger AuthenticateTemp2 = KRSBig.xor(Ns);
         if(AuthenticateTemp1.compareTo(alpha)==0&&
        AuthenticateTemp2.compareTo(beta)==0) {
                System.out.println("Authenticated");
        }
        else{
                System.out.println("Not Authenticated");
        }
}
```