Secure Heterogeneous Cluster-based Newscast Protocol

By

Irum Kazmi

2008-NUST-MS PHD-CSE(E)-20.

MS-08 (SE)

In fulfillment of the requirements for the degree of

MS (Computer Software Engineering)

Thesis Supervisor

Brig. Dr. Muhammad Younus Javed

College of Electrical & Mechanical Engineering

National University of Sciences & Technology

2011

Secure Heterogeneous Cluster-based Newscast Protocol

By

Irum Kazmi

2008-NUST-MS PHD-CSE(E)-20.

MS-08 (SE)

Submitted to the Department of Computer Engineering

In fulfillment of the requirements for the degree of

Master of Science

In

Computer Software Engineering

Thesis Supervisor

Brig. Dr. Muhammad Younus Javed

College of Electrical & Mechanical Engineering

National University of Sciences & Technology

2011

Declaration

ACKNOWLEDGEMENTS

In the name of Allah, the most merciful and the most beneficent

In today’s tech-savvy world of rapid advancement in science and technology, keeping track of the latest inventions is difficult. For that, an environment conducive for research is a must. I am extremely thankful to Almighty Allah, who blessed me with the capability and good fortune of studying at an institute where dreams come true. My experience, as both a student, and a researcher at NUST, has been an unmemorable experience. I, not only had the best learning environment but I also had the privilege of working under an illustrious, competent and amiable team of research advisors.

Foremost, I am extremely thankful to my patient research supervisor, Brig. Dr. Muhammad Younus Javed, who was available at all times. I admire his dedication and honesty towards both his profession and students. Such compassion for learning and sharing knowledge is hard to find in today’s world.

Col. Dr. Khalid Iqbal not only encouraged me as a student, but helped to further nourish my interest in my field.

Dr. Aasia Khanum has been a source of inspiration for me throughout my project. Dr. Saad Rehman has also been an amicable guide, who would cater to any queries that I would come up with.

My husband’s co-operation deserves special mention here. He has always been by my side, encouraging me and tolerating my absence while I would attend to my project. Moreover, my dear son, Rayyan, has also been very patient.

I am also glad to have a supportive family that has always backed me. Thanks is also due to my honorable teachers and class fellows at NUST, for contributing to a beautiful period in my life.

May Allah (SWT) be our constant Guide and Companion throughout our lives (Ameen).

Dedication

I dedicate my minute effort to my research team at NUST, and to my family, back home.

Abstract

The area of peer-to-peer overlay networks is gaining attention of researchers world-wide, due to its popularity and area of applications. Heterogeneous Cluster based Newscast Protocol (HCNP) is one of the many protocols available to design a peer-to-peer overlay network. The characteristic of HCNP is efficient topology generation by introducing completely non-hierarchical clusters with inherent capability of handling heterogeneity. Heterogeneity reflects differences in physical capabilities related to local resources of network nodes (e.g. RAM, storage space etc.). HCNP is one of gossip based protocols that regularly exchanges and updates its cache view with other neighbor nodes to get freshest nodes list. HCNP is vulnerable to attacks just like other unstructured Gossip based overlay network protocols (e.g. Newscast), and can easily be exploited by a malicious node leading to wrong clusters configuration. To avoid this situation, a malicious node should be either restricted outside the network or its interference should be blocked some way. Security measures (e.g. authorization, authentication, confidentiality and integrity) are mandatory to avoid different types of attacks by malicious nodes on the node cache. This research introduces a new version of HCNP that provides security to the overlay architecture. It secures HCNP architecture by restricting malicious nodes outside the overlay. The Secure-HCNP uses Key Assignment on the basis of Heterogeneous Capabilities (KAHC-RSA), a new version of standard RSA designed specifically for HCNP that has dynamic key configuration capability. Secure-HCNP provides heterogeneous security levels dependent on the nodes capabilities. Secure-HCNP is implemented and tested on a peer-to-peer simulation test bed PEERSIM (a simulation tool for peer-to-peer networks). The experiments, when compared with other researches in the same direction, have shown that Secure-HCNP can not only be used as a replacement of HCNP for secure exchange of protocol related information but also for other gossip based protocols security, without compromising much on efficiency. Similarly, KAHC-RSA can be used as an efficient encryption scheme for any smaller piece of information (e.g. Identification number, capability level etc.). This research is a first step towards implementation of an encryption scheme for security of protocol specific information (a layer below application level) in gossip based protocols. The experiments also demonstrate that regardless of computational complexity of RSA, KAHC-RSA can be used in a densely populated environment with more efficiency as compared to the original RSA and CRT RSA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

Publications

1. Irum Kazmi, Saira Aslam, & M. Y. Javed, “Cluster Based Peers Configuration using HCNP in Peer-to-Peer Overlay Networks”, CICSYN 2010, IEEE.
2. Saira Aslam, Irum Kazmi & M. Y. Javed, “Cluster Based Peers Configuration with Multiple Physical Paramters using HCNP in Peer-to-Peer Overlay Networks”, ICCAIE 2010, IEEE.
3. Irum Kazmi & Syed Fahim Yousaf Bukhari,”PeerSim: An Efficient and Scalable Test bed for Heterogeneous Cluster-based P2P Network Protocols”, UKSim 2011, IEEE.
4. Irum Kazmi, Saira Aslam & Muhammad Younus Javed, “Secure- HCNP”, Journal of Computing, Vol. 3, Issue. 4, May 2011.
5. Saira Aslam, Irum Kazmi & M. Y. Javed, “Heterogeneous Peers Configuration to Enhance Cooperation in Peer-to-Peer Overlay Networks”, Journal of Super Computing (Under Review).

CHAPTER 1: INTRODUCTION

1.1 Motivation

A complete survey of peer to peer networks tells that peer to peer network is basically an overlay network that can be used for searching and file sharing etc. Different types of peer to peer overlay networks are used for music sharing, multi-player games, and for distribution of different mirrored websites workload. Peer to peer systems have become very important part of internet and have a variety of applications for content sharing, distribution e.g. Iso-Hunt and other types of more advanced applications e.g. Internet telephony related applications i.e. Skype, VoIP, Usenet, and Instant messaging.

This area has a lot of room for research related to many aspects e.g. security, cooperation between peers to avoid free riding and QoS related issues. All these issues motivated the author to work in this direction to explore some new facts and solve exiting problems faced by p2p networks.

1.2 Background

The current research is all about security of gossip based protocols. HCNP, is one of them. It is found that no work is done related to security of HCNP. HCNP exploits the major characteristics of P2P networks e.g. autonomy, by sharing its cache with neighboring nodes, without any restriction. This makes nodes prone to different types of cache attacks by malicious node/s. So, the security is mandatory to be implemented for reliable protocol specific information exchange by nodes.

1.3 Methodology

HCNP is the target gossip based protocol for the current research. To provide security to this protocol, a modified form of an existence public cryptographic technique, RSA is designed and implemented on PeerSim test bed. KAHC-RSA is specifically designed to work with HCNP as well as other gossip based protocol. KAHC-RSA is tested for efficiency by making its comparison with other fast implementations of RSA and with the original one. Although, RSA is not still used practically for the security of protocol specific information but it can be used for securing a smaller piece of information, regardless of its computational complexity. This is factually proven in a research that RSA can be used efficiently on mobile phones.

First of all, a variety of researches related to the area of the current research, have been studied in detail and the results are observed. Then, algorithm for the KAHC-RSA is written and tested on PeerSim and finally through simulations and experiments, conclusion is drawn.

1.4 Scope of the Project

The research is strictly restricted to HCNP security. But this can be used for other heterogeneous gossip based protocols e.g. HCGNP. The research considers protocol specific information for encryption, not the data exchanged between nodes for file sharing etc. The research is an initiative towards introducing the concept of protocol architecture level security and implementation of public key cryptographic techniques at this level.

1.5 Summary

Secure-HCNP is a modified version of HCNP which is designed to make HCNP secure from malicious activities. The basic problem with HCNP, that makes it insecure, is availability of nodes physical capability related information to all other nodes i.e. either participant or non-participant in cluster configuration, in the unstructured overlay network. This protocol specific information is stored in the cache associated with each node and can be accessed and tempered by any malicious node. This activity can lead to wrong cluster configuration. If clusters are not correctly configured then the basic property of the protocol is not preserved. The issues to handle this situation are how to provide authentication, confidentiality and integrity. Authentication is already implemented for gossip based protocols [14] but it is a centralized solution and confidentiality and integrity is handled using a modified version of a public key cryptographic technique RSA. Public key cryptography is suitable here to avoid key exchange issue as in symmetric key techniques. Standard RSA is computationally quite expensive. So, it is modified to adjust with HCNP. KAHC-RSA is found approximately five times more efficient than Standard RSA. It is more compatible with HCNP as compared to Standard RSA because it preserves the basic property of the protocol i.e. heterogeneity without compromising on efficiency. The concept of heterogeneous security makes it more appropriate in situations where highly diversified nodes are participant in cluster configuration. Optimal key size works well if uniform security is desired for all nodes instead of heterogeneous security. The important fact is, this technique of KAHC-RSA is not strictly restricted with HCNP protocol. This modified version of RSA can be used for encryption of any smaller piece of information exchanged between nodes e.g. node Identifiers. KAHC-RSA is also suitable for the applications or network architectures, dealing heterogeneity related issues. Moreover, KAHC-RSA is implemented with Java that is considered to be the fastest implementation language for RSA.

CHAPTER 2: PEER-TO-PEER NETWORKS & PROTOCOLS

2.1Peer-to-Peer Overlay Networks

2.1.1 Peer-to-peer Networks

Peer to peer is a type of network in which all nodes are distributed across the network without any controlling authority. This type of network helps in resource sharing by maintaining node’s autonomy. Resources may include processing power, storage space, RAM and network bandwidth etc. There is no central coordination or control. There are types of networks in which semi-centralized coordination is used. In fact, the type of peer to peer network is dependent upon the type of its utilization by the applications. There is no need of a server to keep track of each and everything. However, in semi centralized peer to peer networks, a server or a node among the group of nodes can be used as a super peer to keep track of node indices or to index nodes in the network to make it fast for searching.

Peer to peer network is a type of overlay network that configures itself at the application layer of TCP/IP model. The applications specifically designed for the overlay, run at application layer on the top of the overlay. The lower layers structure remains the same. Overlays usually work on the top of the conventional layer 3 and 4.The physical network topology remains the same as in client server based networks. The peers are completely independent of the physical network topology. The standard Internet Protocol (IP) is used for communication underneath.

The applications designed for the peer to peer overlay networks are usually completely dependent on the underlying topology. For example, Kaaza is an application that uses semi centralized peer to peer network topology, whereas, Gnutella, uses completely decentralized p2p network.

There are two types of p2p networks:

- Structured
- Unstructured

In structured peer-to-peer networks, peers are arranged in following a specific topology. The peers are organized by following a specific algorithm. Whereas in un structured peer to peer networks, the peers organization is haphazard, and follows no specific protocol, algorithm or structure.

2.1.2 Peer-to-Peer Overlay Networks

In this type of peer-to-peer network, all peers who are willing to join the overlay work as network nodes. The links establish between any of the two nodes who know each other, means, if a node knows the location of the other node, a link between these two nodes will be established. There will be a directed edge between these two nodes, from first to the second. The set of such directed edges and links constitute an overlay network on the basis of how the nodes in the overlay network are linked to each other. Two different types are defined for that. One is structured overlay network and the other is unstructured overlay network. In an unstructured overlay graph, the network is configured using arbitrarily established links. These networks are very easy to construct because any node willing to join the network, can copy links from another node unless and until it constructs its own list of links existing in the network. In such type of network, the query is always flooded to all the nodes to find data, which is the main disadvantage because the nodes that do not have data, also receive the message and query may not be resolved. This query flooding creates congestion in the whole network. This type is very common in peer to peer overlay networks as compared to structure peer to peer overlays. In structured overlays, the query is not flooded; instead it is forwarded to a specific node or a group of nodes. This increases search efficiency. But this type of overlay has to maintain proper structure, which costs structure related information overhead.

2.1.3 Advantages and Disadvantages

As resource sharing is the basic purpose of any type of p2p network, so, whenever a peer joins the network, its resources are made available to all other nodes. This causes availability of so many resources. This is not possible in client server environment where resources are limited mostly.

In peer to peer overlays, any node can join the network at any time. This introduces so many security threats to the whole network. Node authentication, data verification or signing, confidentiality of information exchanged and integrity of the data exchanged between nodes, may easily be broken, if proper measures are not taken. This is very serious issue and needs to be handled. All the protocols used to define an overlay topology should have built-in mechanism to avoid such circumstances or make the protocol tolerant of such security problems to some extent. The peer to peer applications running on such overlays also need to have proper signing mechanism to avoid tempering of data exchanged. This is an extra overhead and needs to be handled.

2.2 P2P Overlay Network Protocols

To define the architecture/topology of the overlay, there are many protocols available. Here, only gossip based protocols are discussed because they are the base line for this research work.

2.2.1 Gossip Based Protocols

The concept of such type of protocols that use gossip based communication, is given by Demers et al in late 80's. Gossip protocols are very simple. They have fast convergence, high scalability and are quite robust to any benign failure. Nowadays, gossip protocols are applied in many areas e.g. data aggregation and information dissemination, peer to peer overlay networks and many other fields.

The basic properties of gossip based protocols are as under:

I. They handle inter-process interactions that are periodic and are always between pair of nodes.

II. The size of the information exchanged during these inter process interactions between nodes, should be bounded.

III. The peers interact in such a way that the state of at least one node changes to depict the state of the other node. If a peer pings the other in such a way that states are not exchanged, means no interaction took place between these two peers.

IV. There is no surety of reliability in communication.

V. The interactions are not so frequent, so protocol efficiency is faster, as compared to typical message exchanges.

VI. Peers are always selected randomly. Any peer can be selected from a group of nodes or the whole network.

There are three main categories of gossip protocols:

Information Dissemination protocols: These use gossip to spread information just as in case of Newscast protocol.

Anti-entropy protocols: Such type of protocols calculate differences in the replicas to repair replicated data or information.

Aggregation based Protocols: The basic characteristic of such protocols is that these compute aggregate of the whole network by information sampling at the nodes.

2.2.1.1 Newscast Protocol

The newscast protocol is commonly used for communication in usually agent-based distributed systems. Newscast is one of the gossip-based protocols that makes use of the basic properties of gossip based communication as mentioned above. It designs and maintains an overlay on the basis of periodically selected random nodes from the network. The constructed topology is proven to be very stable and robust [20], [18]. Broadcast and aggregation are the examples of the protocols that are built upon the newscast protocol. In Newscast, neighbor nodes are randomly selected using selectpeer () method. After view exchange process, the views of both nodes are merged by update () method and finally, the views are truncated to adjust the basic view size ‘c’. This truncation is done by observing time descriptors. The node remained in the network is more likely to be truncated as compared to the node joined the network recently as shown in Fig. 2.1. Here, two nodes A and B, are exchanging states to get the freshest list of nodes in the cache. The views are merged and truncated. This procedure is repeated periodically.

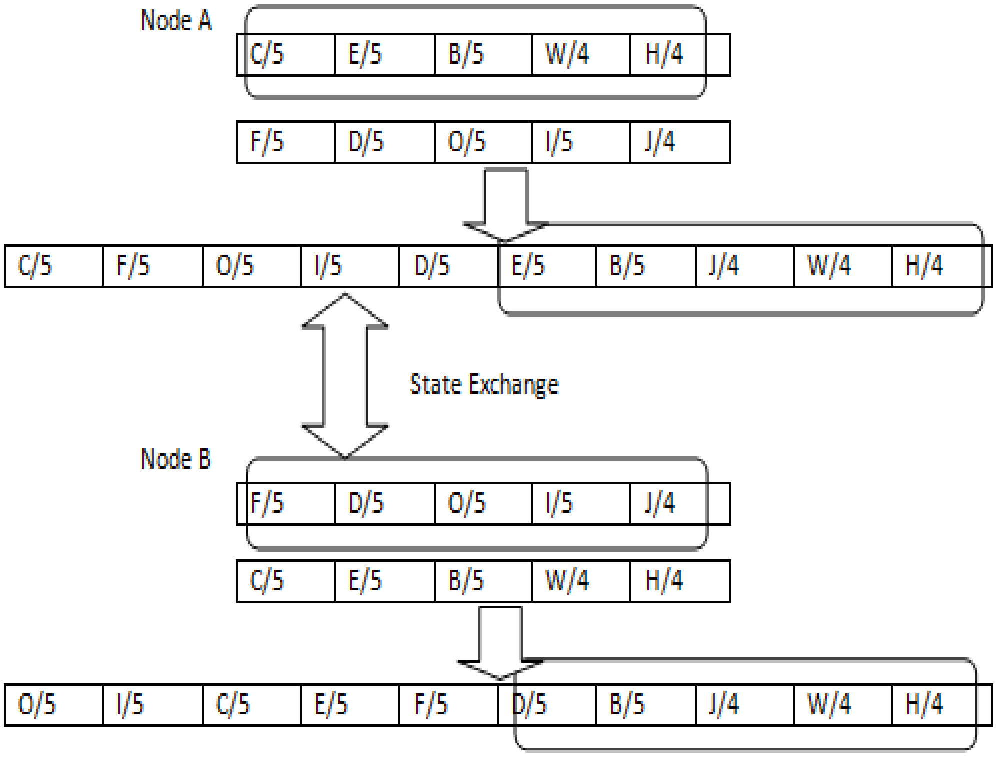


Figure 2.1 Newscast maintaining the updated list of nodes

2.2.2 Security of Gossip Protocols

Such protocols are very popular in designing peer to peer overlay because of its properties of randomness and scalability. But, there is a very little work found in the literature regarding its security. The literature review section shows the detail of the work done on security of gossip based protocols. Each research considered its own attack scenario and discovered the way to handle that scenario. HCNP is a newscast based gossip protocol that does not have any kind of security implemented in it. The attack model considered for the current research is given as under.

2.2.2.1 Attack Model

The attack model for HCNP [2] and [3] is the same as the attack model assumed in [12]. The cache possessed by each node may become polluted easily by single or a group of malicious nodes. Cache pollution means that the cache will contain the ids of either nonexistent peers or the ids of other malicious peers. This will result in the configuration of clusters with nonexistent nodes or clusters consisting malicious nodes. The wrong capability related information will cause misleading clusters thus exploiting the property of the protocol i.e. heterogeneity. The result could be the clusters containing nodes with either the same capabilities or incorrect resource information. This is a serious issue and needed to be considered to prevent the protocol from such attacks. The experiments with Newscast and other gossip based protocols, are shown in literature review portion of the document. The current research handles the above attack scenario by using modified RSA encryption.

2.2.2.2 Protocols Architecture Security

This research is typically related to securing protocol architecture. The information that is specific to protocol e.g. capability level, is encrypted. This is of utmost importance because any p2p application, using such architecture, requires some security at architecture level. This will help the application in terms of security implementation in the application and correct functioning of the protocol underneath.

CHAPTER 3: RELATED WORK

3.1 Literature Review

A complete survey of peer to peer networks [28] tells that P2P network is basically an overlay graph that is designed for searching, sharing of information and to work as a distributed objects store. Several p2p overlay protocols are applied for music sharing, replication of electronic address books, multi-player games, provisioning of mobile, location or ad-hoc services, and the distribution of workloads of mirrored websites [29]. Peer to peer systems have become very important part of the internet and have a variety of applications for content sharing, distribution (e.g. Iso-Hunt) and other types of more advanced applications such as Internet telephony (i.e. Skype, VoIP, Usenet, and Instant messaging).

The basic characteristics of P2P networks are decentralization, autonomy and shared provision of resources and services [21]. As p2p overlays are highly decentralized, so the IP of the peer having the desired content is first located and then those contents are downloaded through another connection [30].

HCNP works on the top of NEWSCAST protocol. The newscast model is one of the gossip based protocols. Such protocols are largely used in agent based systems for the purpose of communication [20].However, these days this category of protocols is used widely to design other p2p networks topology.

The current research is an initiative towards securing one of such protocols, HCNP [2] and [3]. HCNP is a new protocol and it is inevitable to provide security to the nodes. In pure peer to peer networks, it is very important to maintain the autonomy of nodes. In case of [2] and [3], this property is exploited to design clusters. There, the participant nodes need some architecture specific security implementation to prevent them from malicious attacks. HCNP works a level below application layer. So, some security is required at this level to make the protocol specific information secure.

The current research uses a modified version of original RSA. The modified version is designed in a way to work well with the protocol basic properties and still remain efficient.

“RSA cryptography exploits properties and interrelations of numbers, constructed as large powers of huge numbers” [5], RSA uses modular exponentiation technique for key generation which is very popular and the most important operation in public key cryptography [15, 16] but is both complicated and time-consuming. Many researchers are devoted to finding ways to reduce the time such as binary method, signed-digit recoding method, common-multiplicand multiplication method, and high-radix method [5].

The efficiency of RSA is also tested for use on modern mobile phones. For the said purpose, some fast variants of RSA are modified and tested using java based implementation with a usual RSA setup and timing for encryption and decryption. When those are compared for efficiency, it is found that both RSA encryption and decryption can be used efficiently on recent mobile phones [6]. This shows that RSA can be adjusted for efficiency, regardless of its computational complexity.

The strongest known algebraic attacks e.g. factoring, and partial key exposure attacks [7] are collected for a fast variant of RSA.

The attack model for HCNP is the same as described by [12]. According to [12], a malicious node could propagate wrong protocol- specific information by creating fresh timestamps of other colluding malicious nodes. When such poisoned cache is sent to the other non-malicious neighbouring nodes, their cache also gets poisoned. When this non-malicious node exchanges its cache with another non-malicious node, the other node’s cache also gets polluted. In this way, in very small number of exchanges, the whole network can become polluted. Eventually, these non-malicious nodes cache will contain fake Ids of non-existent peers. This issue is handled by [12] by using Secure-Peer Sampling Service. It is implemented on Newscast. It drops malicious nodes and leaves the network partitioned. Secure-PSS is also vulnerable to attacks [12]. In case of HCNP, capability level information can be exchanged wrongly. The technique given by [12] cannot be implemented because it works on the basis of Trusted Prompt concept which is centralized to some extent rather them being completely decentralized. Secondly, it works on the basis of IPs. What if a peer is not accessible due to firewall etc. Moreover, the physical capability related information is more critical and cannot be distinguished from peer to peer. So, encrypting the potential information is a better idea. The certification Authority is not the part of the protocol [14] and does not cost overheads. It will be required before joining an overlay. [13] gives the idea of SPSS with mosquito attack handling. It avoided CA and cryptography without mentioning any reason. Also, [13] handled particularly two groups with different roles and sizes.

Another research suggests some design directions to prevent P2P networks from worm attacks or make the network resilient to worm attacks [17].

The solutions proposed by [23] and [24] can defend only if the cache is < 75 percent polluted whereas in another research on Puppetcast (a gossip based protocol using peer sampling service), the proposed protocol is found highly resistant to attacks by malicious nodes. PuppetCast continues to work even when 50% (or more) of the nodes in the system are malicious and attacking [22]. The attack model supposed by [22] is almost the same as considered in the current research i.e. protocol messages contain malicious contents or they are stopped to be accessed in some way. PuppetCast in contrast, is not 100% decentralized. It needs a central trusted authority. The Brahms [25] protocol is found to resist even when 20% of the nodes in the system are malicious. The Brahms protocol is also fully decentralized [25].

None of the protocols with secure peer sampling service consider implementing any type of encryption technique. Similarly, neither of them gave 100% decentralized secure solution for securing gossip based protocols. If the proposed solution is decentralised, as in case of Brahms [25], it resists only 20% of malicious nodes. The current research implements public key encryption technique (RSA), with dynamic key assignment characteristic, which provides security by incorporating heterogeneity i.e. the basic property of this protocol. This concept of heterogeneous security is not implemented in any of the gossip-based protocols.

An extensive experimental evaluation by the authors in [26] shows that their proposed solution is efficient in dealing with a large proportion of malicious nodes. They, specifically, experimented on Newscast and used public key cryptography but did not mention any details of the type of the encryption technique they used and its its implementation. Similarly, there was no information provided about the overheads caused by the encryption technique they used.

RSA is not yet considered for encryption of protocol architecture-related data. In most recent researches, RSA is most widely used for encryption of application related data e.g.

text files, sequence of characters, integer strings as in [32], [33], [34] and [35] respectively. File sizes used in these researches are 394 to 4933 bytes [32], 128 bit to 10 k [33], 1-10,000 characters [34], and 547 bytes to 2188 bytes [35], which are larger in length as compared to protocol-related data.

Hence, the current research is not only a new idea to implement encryption techniques, KAHC-RSA in this case, at protocol architecture level. It not only prevents protocol specific information from being tempered, but also introduces a new version of a public key cryptographic technique, RSA, to handle smaller pieces of heterogeneous information more securely and more efficiently than standard RSA. The new technique can be tested for architecture security of any gossip-based protocol. The research uses the fastest implementation of RSA using Java. As Java is an object-oriented, cross-platform language and also provides lots of class libraries which are implemented with native programming language, its execution efficiency is very high [1].

3.2 Previous work (HCNP)

The current research is based on a peer to peer overlay network protocol known as Heterogeneous Cluster-based Newscast Protocol. HCNP [2] and [3] does not have any built-in mechanism to protect its architecture against any malicious activity by any of the nodes. Consequently, by introducing any malicious activity in the cluster, the basic properties of the protocol can be exploited very easily. This may lead nodes to sufferance from attacks like misleading cluster joining, wrong cluster formation and wrong information dissemination etc. The HCNP protocol may fail to work or the delays may exceed to infinity. This is a serious issue to be handled at this stage. At least some primary level security should be embedded in the protocol to protect cluster participants from architecture destruction attacks. More sophisticated security may be provided at later stages or might be embedded in the application layer protocols.

3.2.1 Heterogeneous Cluster based Newscast Protocol

HCNP is very close to [9] in terms of emphasis on optimal capacity-proportional distributions of random walks, except, it incorporates heterogeneous clusters. In another research, a scalable and quite simple mechanism for building random overlays and performing random selection with the concept of heterogeneity is studied [11]. However, this does not introduce the concept of clusters as in case of HCNP [2] and [3]. HCNP works on the basis of the idea that the nodes high in resources such as capacity or storage space should be selected more often than the other nodes as proposed in [8]. HCNP does not incorporate any QoS-aware mechanism as suggested by [10]. The clusters configured using HCNP are not quality aware.

HCNP focuses on P2P overlay networks, dealing with heterogeneity of the network participants. In P2P overlay networks, the nodes connected together are enriched in physical resources to be shared among the nodes other than files and applications like RAM, Extra storage, Processor cycles, etc. The nodes are connected with each other, normally on the basis of the contents to be shared. However, the physical resources of the nodes sometimes limit the content based file sharing, and other applications. The particular content may be available but its physical resources restrict the ability of sharing that content efficiently. Decreasing costs for the increasing availability of these resources where the usage of the internet is also increased have created new directions for P2P network-based applications. This resulted in a rapid increase in the development of peer to peer applications in these directions along with controversial discussions regarding limits and performance and the economic, social, and legal implications of such applications. Moreover, with the rapid advancement in other fields of networks, P2P networks also require a study in step ahead. There are many ways to improve the functionality of P2P overlay networks. HCNP is **a non-hierarchical cluster-based** P2P overlay architecture. The main reason to introduce clustered approach is to deal with the heterogeneity of the physical resources, i.e. handling of nodes in the network with different storage capacity, RAM and processor speed etc. Physical resources such as available bandwidth of a node are out of the scope of this research, as currently the issues related to the overlay topology have been discussed.

HCNP uses a clustered approach to design the topology of a P2P overlay network. These clusters are implemented using Newscast protocol which adopts a random approach to generate nodes to form an overlay P2P network. Physical parameters are the basis on which clusters are designed on the top of Newscast protocol and then the behaviour of cluster-based overlay P2P networks is observed and conclusions are drawn. Hence, instead of using pure Newscast as the protocol for defining overlay graph; a Heterogeneous Cluster-based Newscast Protocol (HCNP) has been developed for overlay topology generation and maintenance. Such overlay architecture will be beneficial for applications needing high resources e.g. audio, video or data sharing applications.

3.2.1.1 HCNP Components

HCNP is comprised of the following two components:

- Peers capability exchange and capability level assignment
- Clusters-based peers configuration

These two components of HCNP deal with measuring or calculating the node’s capability level values and assigning cluster-ID according to that capability level. Therefore, each node will have to maintain the data structure as shown in Figure 3.1

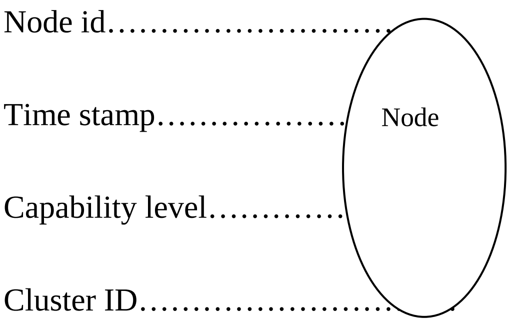


Figure. 3.1 Node’s Data Structure

Each node also maintains a cache of neighbouring nodes, as shown in Fig. 3.2; each node is identified by its descriptor (id, time stamp, Capability Level, Cluster Id) quite similar to the descriptor of Newscast Protocol. Every node maintains its data structure at the first index of the cache.

1/0/3/2	2/5/5/3	6/1/2/5	8/1/3/2	10/2/5/2

Figure. 3.2. Node’s Cache

A node's cache view (n neighbour nodes in the cache). For example, in **1/0/3/2,** 1 = Node Id, 0 = Time Stamp, 3 = Capability Level, 2 = Cluster Id

3.2.1.2 HCNP Architecture and Algorithm

This section enhances the description of HCNP by elaborating its development in order to achieve efficient environment through clusters, in heterogeneous P2P overlay networks. To develop HCNP over NP, the very first step is to design clusters on the basis of capability levels exchanged between nodes. These clusters are designed on architecture-based topology. The clusters are formed so as to attain the maximum level of efficiency in the context of sharing a node's physical parameters. These capability levels are based on purely physical parameters, where resources of the node, such as, **Node's secondary storage, RAM and Processor speed** are the main parameters that should be considered**.** It is performed in a way to ensure heterogeneity of the node's capability, and to increase cooperation as well. The layered architecture of HCNP is shown in Figure 3.3.

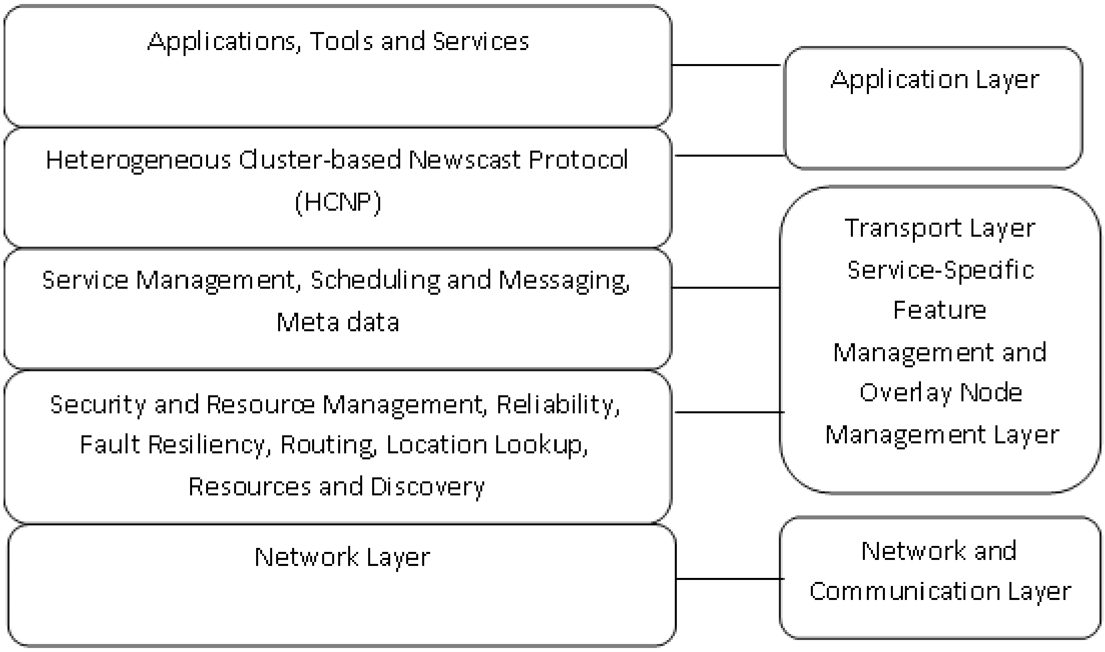

Applications, Tools and Services
Heterogeneous Cluster-based Newscast Protocol (HCNP)
Service Management, Scheduling and Messaging, Meta data
Security and Resource Management, Reliability, Fault Resiliency, Routing, Location Lookup, Resources and Discovery
Network Layer
Application Layer
Transport Layer Service-Specific Feature Management and Overlay Node Management Layer
Network and Communication Layer

Figure 3.3 Layered Architecture of HCNP

The following steps are taken in cluster designing:

Peers Capability Exchange and Capability Level Assignment

Each node will be assigned a distinct "Level ID" on the basis of its physical parameter (free storage space in this case) exchanged. Different levels of the physical parameter are termed as capability levels, which are extracted implicitly from the node, where the node is unaware of the whole procedure. The capability level measurement procedure is embedded in HCNP. There will be a total of ‘N’ levels showing different capabilities where each node will have only one particular level at one time instant.

Each node has to maintain the updated list of ‘c’ neighbors (a fixed parameter). This is achieved by using the Newscast protocol. Each node maintains its data structure as mentioned above, including the capability levels of neighboring nodes as well as its own. After updating the list of ‘c’ neighbors each node compares its capability level with all its neighboring nodes.

A node exchanges its view with its neighboring nodes (depending on the cache size) and assigns a capability level to it on the basis of the extracted value of its physical parameter at that particular point in time. This capability level of the node is used for comparison and assignment of cluster Id.

The algorithm for capability measurement and assignment of levels is as under:

```
$N \leftarrow$ node
$N_C \leftarrow$ cache nodes
$N \rightarrow$ extract physical parameters of $N_C$
$N \rightarrow$ Assign Cap_Levels to $N_C$
```

Each node willing to join the cluster will show its level and not the original parameters to ensure security of the node’s physical resources information. The total number of distinct capability levels handled by HCNP, are defined by a threshold value. Threshold value defines the total number of nodes with homogeneous capability levels allowed in a cluster. Larger threshold values depict decreased number of levels and less heterogeneous clusters and vice versa. This value is adjustable according to the required level of heterogeneity.

Cluster-Based Peers Configuration

On the basis of capability measurement and capability level assignment algorithm, each node (peer) can be configured to become a part of a particular cluster. The following steps are followed to assign cluster ID to a node:

- Every node maintains a cache where it resides the information related to the neighboring nodes. At the first index of the cache the node maintains its own data structure values i.e. its node ID, time stamp, capability level and cluster ID (which is initially set to 0).
- Each node joining the network configures its cache nodes on the basis of predefined capability levels arranged in ascending and descending order. A cluster is composed of heterogeneous capability levels.
- Every node assigns a Cluster ID (0 for non-participant and an integral value for participant node) to the nodes (node IDs) residing in the cache of a particular node showing its participation to a cluster. This is maintained by each node’s data structure and will be accessible to every node in the suburbs of a node.
- The size *(s)* of a cluster can be configured dynamically according to the cache size which is again dependent on the network size.
- The cluster size is less than or equal to the cache size to avoid cache-miss (occurs when a node is not found in the cache), conditionally the degree (size of views exchanged between the peers) is half of the cache size. As *peer-sampling-service* (PSS) is used in NP, the cluster size is restricted to one sample size.
- Each node in the cache maintains only one Cluster ID at a time in its data structure to avoid multiple cluster participation.
- The most important point is that one cluster can never contain all nodes of the same capability level, ensuring heterogeneity.
- Each node will refresh both its neighbor’s list as well as its capability level after each cycle, which shows that there could be a change in the capability level of a node after a specific period of time.
- The total number of active/ non-empty clusters formed is less than the total number of clusters formed in a cycle, due to a rapid change in data structure (Cluster ID) of a node on the basis of rapid changes in its capability level.
- The overheads (Cluster ID & Capability level) are handled implicitly by maintaining the updated data structure in each cycle.

  The Algorithm for assignment of Cluster ID to a node on the basis of its capability parameters is as under:

```
for int  i = 1 to cache length
{
N → check capability level of $N_C$
        If
        Current cluster size < = cache length &
        cluster has not more than ‘n’ nodes of same capability  level
        N →Assign Cluster ID to $N_C$
        else
        Calculate next cluster ID
        N →Assign next cluster ID to $N_C$
}
```

Here n= Threshold value (larger value of n shows reduced heterogeneity and vice versa). Hence after performing the configuration on its cache, each node maintains a set of clusters based on the above conditions discussed earlier. The number of clusters depends upon the size of the network and the length of the cache associated with it. The cache length can be configured according to the network size. The rate of change of cluster ID associated with a node depends on the rate of change in the capability levels, associated with each physical parameter of the node.

3.2.2 HCNP with Single Physical Parameter

The algorithm associated with HCNP is tested on PEERSIM simulator [4]. On the basis of observations in the output files, it can be seen that as the node’s capability level changes, it is assigned a new cluster ID provided that changed capability level is not required in the previous cluster. At each cycle, the capability level is picked and clusters are adjusted accordingly, as shown in Figure 3.4. In Figure 3.4 it can also be seen that each cluster shows the number of nodes with different capability levels. Hence it would be easier for the node to locate its place according to its own capability level, e.g. in cluster ID: 6, the number of nodes with the lowest capability level is a maximum, and the number of nodes with the highest capability level is a minimum. The new node joining the network will, therefore, not become a part of cluster ID: 6, it may become a part of cluster ID :1 or cluster ID: 9.

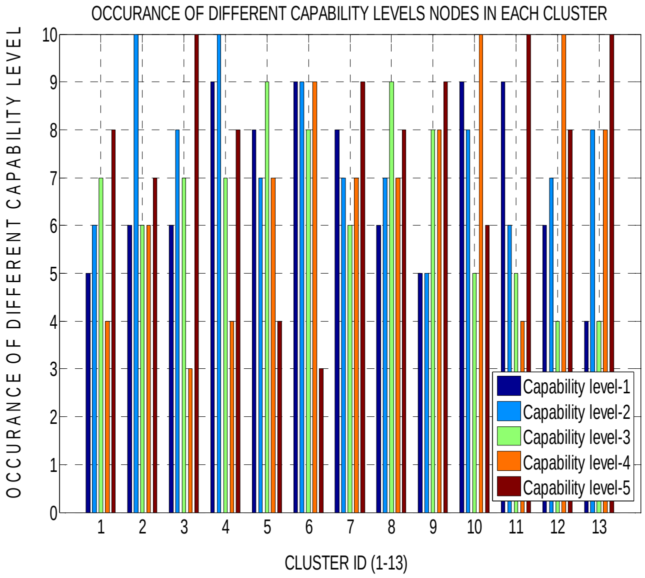


Figure 3.4 Cluster having Different Capability Levels of Nodes

The simulations further explain that the total number of clusters developed at any instant of time in one cycle depends upon the size of cluster configured according to the network size and length of the cache associated to a node. In Figure 3.5 it can be seen that in a network of 8000 nodes, the total number of clusters developed at any instant of time (t) is approximately 115, where the cluster size is configured as 30 nodes. Hence, the number of clusters formed in a network at any instance of time *t*, is given as:

No. of clusters (t) = network size/cluster size

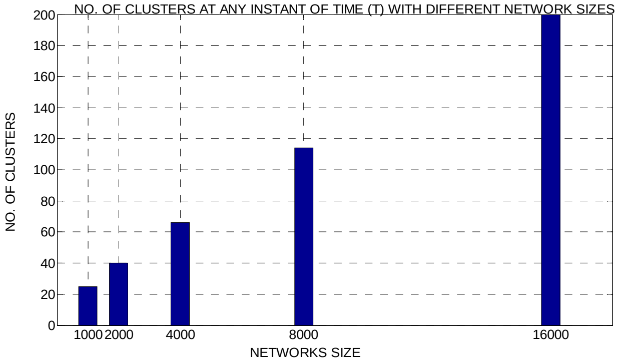


Figure 3.5 Number of Clusters Formed in a Network at any Instant of Time (t)

It can be useful at the application level when there is a need to observe a network state. A network state provides information related to specific nodes, i.e. the current cluster ID and capability level of a node. In addition, a cluster state can also be maintained by observing the behavior of a particular cluster in the network, e.g. the vacancy for a new node with capability levels etc.

3.2.3 HCNP with Multiple Physical Parameters

Only level assignment step differs in case of multiple physical parameters. An integral capability level is calculated for the said purpose. The following steps show, how a capability level is assigned to a node in case of multiple physical parameters:

- Assign “P” for physical parameters
- Assign “ L” for cap levels
- Total possible combinations are LP, where L is a constant value for all the nodes in the network.
- L * P = max no. of capability levels

Suppose

Physical Parameter= P_m

Where ‘m’ is from 1->n

Capability Level=L

Where L is from 1->k

Then,

ICL= $\sum_{m=1}^{n} P_m$

Eq. 3.1

Where Value of P_m =L

Equation 3.1 shows that the cluster is comprised of heterogeneous nodes having diverse physical parameters to provide maximum utility out of each node’s capability. An integral capability level (ICL) is assigned to a node which is the sum of all capability levels (L). These Capability Levels are assigned to the nodes on the basis of their physical parameter, e.g. if there are 5 ‘L’ to be assigned, then the capability level for 3.0 GB RAM is 2, that for 80 GB secondary storage is 3 and capability level for 1000 MHZ processor speed is 4. The ICL would then be 9 (2+3+4=9), hence considering all possible combinations of these capability levels, ICL could be assigned to each node.

On the basis of capability measurement and capability level assignment algorithm, each node (peer) can be configured to become a part of a particular cluster. The following steps are followed to assign cluster ID to a node:

- Every node maintains a cache where it resides the information related to the neighbouring nodes. At the first index of the cache, the node maintains its own data structure values i.e. its node ID, time stamp, capability level and cluster ID (which is initially set to 0).
- Each node joining the network configures its cache nodes on the basis of predefined capability levels arranged in ascending and descending order. A cluster is composed of heterogeneous capability levels.
- Every node assigns a Cluster ID (0 for nonparticipant and an integral value for participant node) to the nodes (node IDs) residing in the cache of a particular node showing its participation to a cluster. This is maintained by each node's data structure and will be accessible to every node in the suburbs of a node.
- The size 's' of a cluster can be configured dynamically according to the cache size which is again dependable on the network size.
- The cluster size is less than or equal to the cache size to avoid cache-miss (occurs when a node is not found in the cache), conditionally the degree (size of views exchanged between the peers) is half of the cache size. As peer-sampling-service (PSS) is used in NP, the cluster size is restricted to one sample size.
- Each node in the cache maintains only one Cluster ID at a time in its data structure to avoid multiple cluster participation.
- The most important point is that one cluster can never contain all the nodes of the same capability level ensuring heterogeneity.
- Each node will refresh its neighbour's list and its capability level after each cycle, which shows that there could be a change in the capability level of a node after a specific period of time.
- Total number of active/ non-empty clusters formed is less than the total number of clusters formed in a cycle, due to rapid change in data structure (Cluster ID) of a node on the basis of rapid changes in its capability level.
- The overheads (Cluster ID & Capability level) are handled implicitly by maintaining the updated data structure in each cycle.

  The Algorithm for assignment of Cluster ID to a node on the basis of its capability parameters is the same as in HCNP with single physical parameter:

CHAPTER 4: SECURE-HCNP

4.1 Secure-HCNP Overview

HCNP is one of many gossip based protocols that are used to design an efficient peer-to-peer overlay network. The specialty of HCNP is to design an architecture comprised of heterogeneous nodes uniformly distributed across a cluster. The nodes are heterogeneous in their physical resources such as RAM, secondary storage and processor speed etc. The benefit of such architecture is basically to provide an efficient, scalable and self-configuring overlay topology with a built-in mechanism to configure clusters on the basis of nodes capabilities. The purpose of heterogeneous clusters is to provide architecture-level resource sharing. This topology is quite helpful for the applications involving file search. The resources (e.g. disk space) can easily be shared among the nodes of a cluster. A node which is deficient in a resource can make use of the resources available to the other node, if required. The architecture suggested and tested by HCNP acts as a backbone for the peer-to-peer applications and provides robust resources access to nodes. HCNP is Newscast [20] based and hence, preserves all the properties offered by Newscast Protocol.

The Secure-HCNP is an extension of HCNP. HCNP does not have any built in mechanism to provide security. The architecture specific information shared among nodes may be accessed and altered by any malicious node. HCNP is prone to attacks. Secure-HCNP is the first step towards providing security to the architecture so that at least, the information related to HCNP architecture is preserved. Secure-HCNP is typically based on KAHC-RSA, a new version of standard RSA encryption algorithm, having property of dynamic key assignment on the basis of heterogeneous capabilities of nodes. This not only retains basic HCNP properties but also decreases standard RSA overheads. The research suggests an optimal key size that can be used uniformly across a cluster. The optimal key size not only ensures confidentiality of architecture specific information exchanged between nodes but also reflects cluster wise need of heterogenic security level. The heterogeneous security is to maintain HCNP basic property i.e.

heterogeneity and to provide each node with the optimal security level based on its physical capability. This mechanism also reduces standard RSA computational overheads. The network size does not impact on the efficiency of the protocol. Two major aspects related to security are covered by secure-HCNP. One is confidentiality and the other is integrity. The third major characteristic i.e. authentication is already implemented on the gossip-based protocols. The issue of detection of nodes not behaving according to the prescribed protocol and their removal from the network always requires a central database maintaining blacklisted nodes [14].

4.2 Secure-HCNP Characteristics

Secure-HCNP has the following basic characteristics:

- To ensure confidentiality, KAHC-RSA is used which is a modified version of standard RSA.
- MD-5 hashing algorithm is used to maintain system integrity and prevent the architecture from attacks in the presence of malicious peers in the network. The purpose is also to reduce the size of the message for fast transportation. Each node maintains a security identifier depicting its participation level (true/false)
- One of the problems is how to ensure the identity of a node. Authentication can be achieved through public key cryptography and a trusted certificate authority (CA) [14].CA can not be a performance bottleneck because it does not participate in the protocol and it can even be offline.

4.3 Secure-HCNP Architecture and Layers

Secure HCNP resides under the application layer and above the transport layer of standard TCP/IP model. Figure 4.1 presents its position in the TCP/IP model.

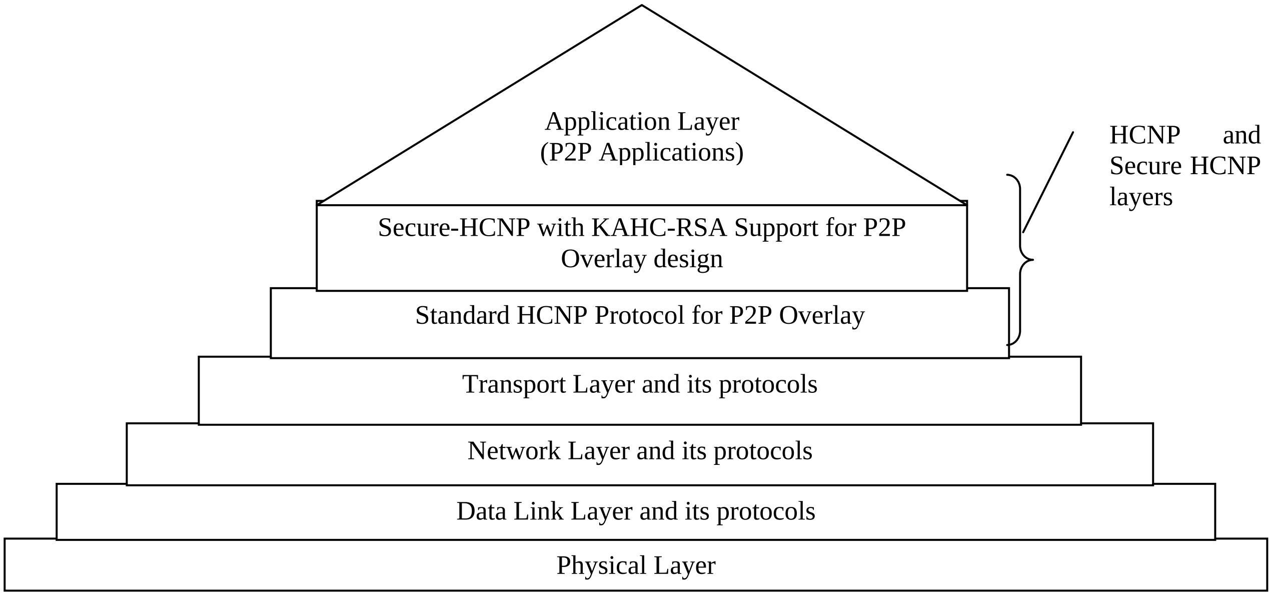

Application Layer
(P2P Applications)
Secure-HCNP with KAHC-RSA Support for P2P Overlay design
Standard HCNP Protocol for P2P Overlay
Transport Layer and its protocols
Network Layer and its protocols
Data Link Layer and its protocols
Physical Layer
HCNP and Secure HCNP layers

Figure 4.1 TCP/IP Layers Showing HCNP and Secure-HCNP Layers

Secure-HCNP resides on the top of standard HCNP protocol and can be used by the applications running on the top of it. All applications using HCNP protocol for defining overlay architecture can replace it with Secure HCNP to make underlying architecture secure. The applications do not have to compromise much on efficiency.

4.3.1 Standard HCNP Protocol Layer:

It consists of standard HCNP Protocol that has inherent capability of Cluster-based heterogeneous resource sharing. It is used to design a p2p overlay graph on the top of which a p2p application runs. HCNP is discussed in detail above.

4.3.2 Secure-HCNP Protocol Layer:

This layer ensures secure transmission of HCNP architecture-specific information. This leads to the correct configuration of HCNP clusters. The applications do not need to embed built-in security in it. This ***partial implementation of security*** also increases the efficiency of application as compared to the applications having complete built-in security implementations.

4.4 Standard RSA Implementation on HCNP

Standard RSA algorithm is a public key encryption algorithm which is used for encrypting a smaller piece of information that needs higher levels of security. It is computationally quite slow as compared to symmetric key cryptographic techniques and other public key encryption techniques. 1024 bit RSA gives security equivalent to 128 bit AES (a symmetric key encryption technique). The reason to choose RSA is that it does not require any key exchange procedure as AES, DES etc. require. HCNP is completely decentralized architecture and does not have any central authority to ensure secure key exchange procedure. It is computationally slow but not bad for exchanging smaller piece of information as in the case of HCNP where only physical capability related information is required to be exchanged between nodes.

4.4.1 RSA Algorithm

Basic algorithm of Standard RSA is given as under:

There are three basic steps:

1- To generate keys

2- To encrypt the message

3- To calculate digital signatures

In *key generation* procedure, the very first step is to select two prime numbers randomly e.g. ‘a’ and ‘b’ in such a way that ‘a’ should not be equal to ‘b’. Then their modulus is computed as n=a*b. After modulus calculation, Phi is calculated as $= (a - 1)(b - 1)$. Afterwards, a public exponent is selected in such a way that it is less than and greater than 1 and gcd(e,) should be 1. Finally, $d = e^{-1} \bmod$ is calculated**.** Here, (n, d) will serve as private key and public key will be (n,e). In *encryption* phase, the following procedure is done: $c = m^e \bmod n$, decryption: $m = c^d \bmod n$. Then the last step is to compute digital signature. For that purpose use the following procedure. $Y = H(m)^d \bmod n$, verification: $m' = Y^e \bmod n$, if $m' = H(m)$ signature is correct. H depicts a hash function.

4.4.2 RSA Usage

RSA is used in many security protocols e.g. PGP, IPSEC, SSH and SSL etc. Many other implementations and usages of RSA are found in literature.

4.4.3 Standard RSA Implementation on HCNP Examples:

All Prime numbers mentioned below are in decimal format and can be converted to binary following the standard decimal to binary conversion method.

4-bit key:

Message = Capability Level= 2 (Ranging from 1-5)

RSA bits = 4

Prime Numbers: p= 11, q= 13

Binary format: p=11=1011, q=13=1101

Modulus N=p*q= 143

PHI= (p-1)*(q-1) = 20

Public Key = 65537 (generic key is used for experimentation)

Private Key = 113

Encrypted Value= 84

Hashed (Sender side) = 9Px09zs97BvRZB7I9cvQnA

Hashed (Receiver side) = 9Px09zs97BvRZB7I9cvQnA

Un-Hashed = 84

Decrypted Value= 2

8-bit key:

Message = Capability Level= 2 (Ranging from 1-5)

RSA bits = 8

Prime Numbers: p= 199, q= 179

Modulus N=p*q= 35621

PHI= (p-1)*(q-1) = 35244

Public Key = 65537 (generic key is used for experimentation)

Private Key = 7325

Encrypted Value= 22985

Hashed (Sender side) = lK3AwFv34fgldEJRh45GtQ

Hashed (Receiver side) = lK3AwFv34fgldEJRh45GtQ

Un-Hashed = 22985

Decrypted Value= 2

***32-bit key*:**

Message = Capability Level= 4 (Ranging from 1-5)

RSA bits = 32

Prime Numbers: p= 3574654261, q= 3219421103

Modulus N=p*q= 11508317363792269883

PHI= (p-1)*(q-1)= 11508317356998194520

Public Key = 65537 (generic key is used for experimentation)

Private Key = 10059614819572997993

Encrypted Value= 8540222292004936118

Hashed (Sender side) = 3TIhS1F62QhJOnqWL6/rvg==

Hashed (Receiver side) = 3TIhS1F62QhJOnqWL6/rvg==

Un-Hashed = 8540222292004936118

Decrypted Value= 4

64-bit key:

Message   = Capability Level= 5 (Ranging from 1-5)

RSA bits = 64

Prime Numbers: p= 17960315784160238351, q =   12280551836681366809

Modulus N=p*q=   220562588990546357854056256785900291959

PHI=(p-1)*(q-1)=   220562588990546357823815389165058686800

Public key = 65537 (generic key is used for experimentation)

Private key = 90537784290151793920629287262438146273

Encrypted Value=   213531348473650248320001238533207768943

Hashed (Sender side) = m/DWvESFf75gyCIfXPZa5w==

Hashed (Receiver side) = m/DWvESFf75gyCIfXPZa5w==

Un-Hashed = 213531348473650248320001238533207768943

Decrypted Value= 5

***128-bit key*:**

Message   = Capability Level= 3 (Ranging from 1-5)

RSA bits = 128

Prime Numbers:

p= 179523711197359786574182004961688360397

q= 202755979532338453339980130046634393311

Modulus=p*q=36399505913101320498735109688833014519832773427256320525740840095364414104467

PHI= (p-1)*(q-1) =

3639950591310132049873510968883301451945049373652662228582667796035609135 0760

Public key = 65537 (generic key is used for experimentation)

Private key=

689922734413452863627092379197527665838555370546612908791276673670664459 4033

Encrypted Value=

2807421755212650071664689113702068158319643876891538556022355545671529145 6653

Hashed (Sender side) = WSDtRk3+EjsLVpcvHdKKmQ==

Hashed (Receiver side)= WSDtRk3+EjsLVpcvHdKKmQ==

Un-Hashed=

2807421755212650071664689113702068158319643876891538556022355545671529145 6653

Decrypted Value= 3

256-bit key:

Message = Capability Level= 3 (Ranging from 1-5)

RSA bits = 256

Prime Numbers:

p=

727920785051886270807753528469007013776895979840703300644099562951304041 73553

q=887330203128467162169408532712578470933990249590396353058097707686697 74232147

Modulus N=p*q =

6459060980615235277485007564744340930828153271745457670790736930515772609933402334959684498555923750896508146688189142526008255666626213469279224699808291

PHI= (p-1)*(q-1) =

6459060980615235277485007564744340930828153271745457670790736930515772609933240809860866463212626034690389988139718053903065145701255993742215424521402592

Public key = 65537 (generic key is used for experimentation)

Private key =

694030966103002682821145662311818496954267899654111615083210544649466266676074305858373462836921319808500942925063621093205132307372090695831072821150145

Encrypted Value =

6236031804779084506976848839882062471277037682379352692026132084478483687297019059580137599703693863083719683703209418065332518699863212039917502988535794

Hashed (Sender side) = WV/qNZoi47II76zvZ5Y6cw==

Hashed (Receiver side) = WV/qNZoi47II76zvZ5Y6cw==

Un-Hashed =

6236031804779084506976848839882062471277037682379352692026132084478483687297019059580137599703693863083719683703209418065332518699863212039917502988535794

Decrypted Value= 3

<u>512-bit key:</u>

Message = Capability Level= 5 (Ranging from 1-5)

RSA bits = 512

Prime Numbers:

p=76318213690641862640801336672405650835870908224676907478480568555030298399488979273221238247564098057050069614808238351232899580111160276534920307602109 31

q=120824127149719375479131271759463236295002897492606027644851317304163559607503846982226862117042813682599881518459227975493277588176995100730979903712195 81

Modulus N=p*q=

922108155479756641467452189130083738851110354735686940976234067275448576797862158845824593208485095932957484338797443549069858881158013493642077942386170680098320387962671160707229349629401053561777670493396662808365100839573750998116007451594319811425323355336382823523575075008728328471258777733032774399 11

PHI=(p-1)*(q-1)=

922108155479756641467452189130083738851110354735686940976234067275448576797862158845824593208485095932957484338797443549069858881158013493642077942386170482955979547601433040774620917760513922687971953210461539476479241645715744005289752003493955204513583705385249556057248348831560040315881511832821460094 00

Public key = 65537 (generic key is used for experimentation)

Private key =

183585870770707610324966296348159552993412696775733451422218015927034393244404007638743294508206258018115404361698415298659742110278922746922224590571047690031732016068838950761054880982333424832272732128264982637122558936071212105848920215169890710720558466024791128788851739560159839572174832024576260910 73 Modulus

Encrypted Value=

369501769380048266090850126616078393134124592216601624167915327193630744207302818154470141333872762560672280493858906692305061129212419217382079103721703100888142076363708408812719178448089221669244069797860157658098529905003412075163861390591606153887652356422252005605593740345358813873929932578341846492 96

Hashed (Sender side) = YXVlNbzkWC0iob2EJfQ4+w==

Hashed (Receiver side) = YXVlNbzkWC0iob2EJfQ4+w==

Un-Hashed=

369501769380048266090850126616078393134124592216601624167915327193630744
207302818154470141333872762560672280493858906692305061129212419217382079
103721703100888142076363708408812719178448089221669244069797860157658098
529905003412075163861390591606153887652356422252005605593740345358813873
929932578341846492996

Decrypted Value= 5

1024-bit key:

Message = Capability Level= 2 (Ranging from 1-5)

RSA bits = 1024

Prime Numbers:

p=1349914214934862979263964576372767373677945726749841344028556799916
437673159300499033729860139959882762804993639748201699709844291711832800
351246625148046720619458016354750125211017915834723764805082069940138285
852476470872780268505237118259419480176298354466705935501453527161646342
565076220559541743039973

q=156044941459878769995069974160656580743976331277739640395262069481189
726843040302318125019310764361564308876298224898045765631155467563566465
933037208668263580841692862215551934953517095860599928967612665210803688
209791529636554682717486108934956673850088409653033963886144659928827781
048601679745244780245857

Modulus N=p*q=

210647284645368901205434880808100152822275449228991440870046070922100479
567365865720997127867707490792221617555095976076296471663171547702224820
872685218562888250279302553533699602881378414058495165581816985415300334
489669505133501305149138437231435518742045167844580936221029152417728531
023075572052069662798424647055928175721134901441895712070072105529532007
352424079587270808085639486924649140317793898840057318183030407439812273

448631733793417841491827624698179899291615746153861681559121289013468791
859720737016173577811154584724834642325912528215191707614646316980908651
16931161112590391359465415463909518641861

PHI=(p-1)*(q-1)=

210647284645368901205434880808100152822275449228991440870046070922100479
567365865720997127867707490792221617555095976076296471663171547702224820
872685218562888250279302553533699602881378414058495165581816985415300334
489669505133501305149138437231435518742045167844580936221029152417728531
023075572052069662795514283426394525041920237123916378888895562759487112
535031173324459746440477490896477552826084855625586601370111760977337620
038634141039425178648989286535076884680106960450841079913771172060231854
132773385506753837188474579631485130098680256271430167350778675783509657
29332974022116267745787515159122995356032

Public key = 65537 (generic key is used for experimentation)

Private key =

869176781265554698322683376842492688499621999488897347148600920071935115
806446395292308822771647461129324958714147029167525090668703937159705754
923044033199204123480308780178876765967046870160340916987883931153478438
938255147141331201282176727590899689919035877268284736452548993649421690
941912144198249511164120611626495070904009498802958133255033615841745959
865772462737085991614415110368577442288018472897311880960459319216498149
302644985578854033755888778320879673393571940203494893423287613844513755
505814713959082238315871886475520833747121700732718095961023078648346453
5340682110961260240499046110334681789185

Encrypted Value=

927311203560640988445856161047728252748084129424994693120608708722772701
513165371897358712118883062122473397738554237356947030001055515390178207
870510810367205699737124075006676919675977325613209545720166025842679482
731282453465529061370253512460321416560325359574288017792762010801946151
994763901734239848227366773871886016669874627890411819216752200223591137
022949889545179061725053418682902565529220513443625457984195940139826961

379089681966464408890898831651191507123142722983148114148195601373144018
944525211320065410440710005846060215881459297188180476636392789338926504
4941078587881505336545621249059475759800

Hashed (Sender side) = pcIOG0EdfF+pZjGlIMPKVA==

Hashed (Receiver side) = pcIOG0EdfF+pZjGlIMPKVA==

Un-Hashed =
927311203560640988445856161047728252748084129424994693120608708722772701
513165371897358712118883062122473397738554237356947030001055515390178207
870510810367205699737124075006676919675977325613209545720166025842679482
731282453465529061370253512460321416560325359574288017792762010801946151
994763901734239848227366773871886016669874627890411819216752200223591137
022949889545179061725053418682902565529220513443625457984195940139826961
379089681966464408890898831651191507123142722983148114148195601373144018
944525211320065410440710005846060215881459297188180476636392789338926504
4941078587881505336545621249059475759800

Decrypted Value= 3

4.5 KAHC-RSA Implementation on HCNP

4.5.1 Why Secure-HCNP?

KAHC-RSA provides security to the architecture configured by HCNP. HCNP is used to design an overlay efficiently just like many other p2p topology generation protocols. It lacks in security in the sense that the architecture specific information, that is exchanged by nodes mutually, is available to each and every node and can be accessed and tempered by any malicious node at any time. This may cause the protocol properties to be destroyed at maximum. So, a malfunctioning HCNP protocol may cause wrong cluster configuration. Another major problem that a malicious node may create is that, the nodes are sharing their resources related information with each other. If a malicious node gets that information in some way or the other, it can cause critical problems to the resources of other nodes. So, Secure-HCNP comes into play. It not only makes the architecture specific information secure by using KAHC-RSA encryption, but also prevents nodes from misuse of their resources by any malicious node.

4.5.2 Why KAHC-RSA, Why not Standard RSA?

As symmetric key encryption techniques are not suitable for HCNP due to key sharing problem, the public key encryption technique is used to work with HCNP. As RSA is used widely and commonly in internet related protocols, it is considered as suitable for HCNP. As RSA is very high in its computational complexity, it is modified according to the HCNP protocol to match best with the situation. HCNP is very efficient and scalable and KAHC-RSA is designed to meet the purpose of HCNP. The detailed description of KAHC-RSA is given as under:

4.5.3 Basic Functionality and Steps:

KAHC-RSA uses key assignment on the basis of heterogeneous capabilities of resources a node shares. Heterogeneity is the key attribute of HCNP and is required to be maintained in Secure-HCNP. To retain this major property of the protocol along with the other, KAHC-RSA uses dynamic key assignment technique. The protocol major functions are described in steps as under:

1. Determining key size combinations
2. Dynamic allocation of keys
3. Newscast exchanges and updates between nodes
4. Level Assignment
5. Self Cache Encryption
6. Cache decryption
7. Cluster Configuration

4.5.3.1 Determining key size combinations:

The Standard RSA uses a fixed size key i.e. 512, 1024 or 2048 bits depending on the criticality of the problem area and the level of security required. In HCNP, very high security is not required; so this is not practical to use high bit values for keys that cost high in terms of computation. It’s a good idea to change the bit length according to physical capabilities of nodes. The node that is low in its capability does not need to be so much secured. Its risk of being misused is not as much critical as the other with very high capability. The approach is to use combination of key sizes. Key size should change according to physical capability of node. This seems quite compatible with the major property of HCNP i.e. heterogeneity. Heterogeneous key sizes are used according to heterogeneous capabilities. Now the question is, what key sizes should be used for what values of physical capabilities. So, the very first step is, to find the best key size combination that matches with nodes capabilities. If estimated capabilities of nodes participating in the overlay are ranging between x to y, where x and y are integers, then the key sizes allocation may be from x + n to y + n. Here, n can be any constant. This is mentioned in detail as under:

4.5.3.2 Dynamic allocation of keys:

Heterogeneous key sizes are allocated dynamically because nodes are rapidly changing their capabilities. This is quite possible that a node has 20 GB storage space available at one time instant and just after a newscast update, the node is found very low in its capability, say, 10 GB or the node has left the network. So, instead of static allocation of key for all nodes, it is a good idea to use dynamic allocation criteria. What dynamic allocation means? After a short interval of time, when each node exchanges and updates its cache view with the other nodes in its cache, to get freshest list of nodes, some information is exchanged between nodes. This information is stored in encrypted form in the cache. Dynamic allocation means that this encryption will be done using dynamic key sizes. If a node found in the network is low in its capability, it will be encrypted with smaller key size. If a node is high in capability, it is encrypted with higher key size to provide the node with higher security. Node ids are not important. The node encrypted with higher key sizes values may be encrypted with lower key size values just after a short span of time if it becomes low in its capability. As the key size changes dynamically with each Newscast update view method, it makes it very hard to determine the key value in such a short instance of time. This dynamicity makes KAHC-RSA secure enough to prevent malicious node attacks.

The detail of encryption and decryption procedure is given under the heading cache encryption.

4.5.3.3 Optimal Key Size Values:

While using heterogeneous key size values, it is very important to decide what minimum and maximum key sizes are affordable for the network. Affordability or suitability depends on some factors given under the heading factors involved in determining key combinations. The minimum value is taken as ‘x’ and the maximum value is taken as ‘x+(CL-1)’ then the average key value can be calculated using the following equation:

Range = 2^x when $x1 \le X \le x1 + (CL-1)$ Eq. 4.1

(Deduced for base 2 key sizes e.g. 2, 4, 8, … ,1024 etc only)

Where,

x1= An integer showing minimum key size in bits.

CL= Total number of capability levels

For example, for CL=5 (All nodes in the network comprise of maximum 5 different levels depicting their capabilities), there should be 5 different key sizes to encrypt each capability level. If x1=4, then 2^x means 2^4 =16 bits, and x1+ (CL-1) = 4+ (5-1)=8, and 2^8=256 bits . So, X ranges from 16 bits to 256 bits to accommodate 5 CL values. Increase x1 by 1, because it is an integer, and get next key size used for next level.

The above equation is deduced through experimentation. The detail of experiments is given in next chapters under simulations heading.

The optimal key size value can be obtained by simply finding average of the key size values used.

Optimal Key Size K $\approx [\sum_{x1}^{x1+(CL-1)} x1]$ /CL Eq. 4.2

Where,

x1= integer

CL= total number of capability levels.

If x1=8 then x1+1=9, x1+2=10, x1+3=11, x1+4=12 for CL=5. K= 8+9+10+11+12/5=50/5=10 bits. So approximately, 10 can be used as optimal key size for uniform security provision.

Optimal key value can be used for uniform level of security regardless of the capability of the node. This can cost much if network bandwidth is the issue. All nodes will have equal level of security even if they are heterogeneous in their capabilities. E.g. if a node has a physical resource (e.g. 5 GB hard disk space) that can be shared and another node also has a physical resource that is lesser than the previous one (e.g. 0.5 GB) then both will be handled in the same manner. It could be a better approach if higher resource level node is given higher security because it is on higher risk as compared to the node that has almost nothing to share.

Optimal key size is good in the sense that it give an average view if the nodes capabilities in the network. The tendency of nodes about resources is well handled using Optimal Key Value.

4.5.3.4 Newscast exchanges and updates between nodes:

The first step is simple conventional newscast exchange and update procedure between nodes. Nodes get refreshed list of peers through this.

4.5.3.5 Level Assignment:

Nodes extract its physical capability and on the basis of that capability, each node is assigned a level, just as in standard HCNP. This level information is stored in the cache.

4.5.3.6 Self Cache Encryption:

Each node picks its capability level. The protocol assigns number of bits to calculate keys and sets the security Id value to 1, showing that the node is now using KAHC-RSA encryption. For example, its capability level is 5 (very low in capability let’s say), it will be encrypted using any public key. After encryption, it will be hashed using MD-5 hashing. This makes the length of the message short for transportation. After encryption, the node will save encrypted hashed information in its cache. This information is accessible for other peers/ nodes when required. The flow chart of the encryption procedure is shown in Figure 4.2.

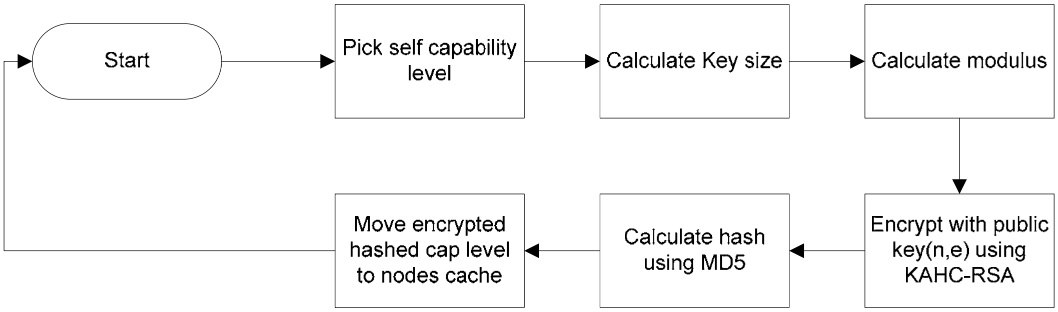

Start
Pick self capability level
Calculate Key size
Calculate modulus
Encrypt with public key(n,e) using KAHC-RSA
Calculate hash using MD5
Move encrypted hashed cap level to nodes cache

Figure 4.2 Encryption using KAHC-RSA

4.5.3.7 Cache decryption:

When a node refreshes its cache using simple newscast protocol, it also extracts the other node's physical capability related information in encrypted format. It then decrypts it using its own private key. This is the key that is calculated using dynamic key size allocation procedure. The encrypted message is first hashed and checked for integrity and then decrypted. The decryption procedure is shown in Figure 4.3.

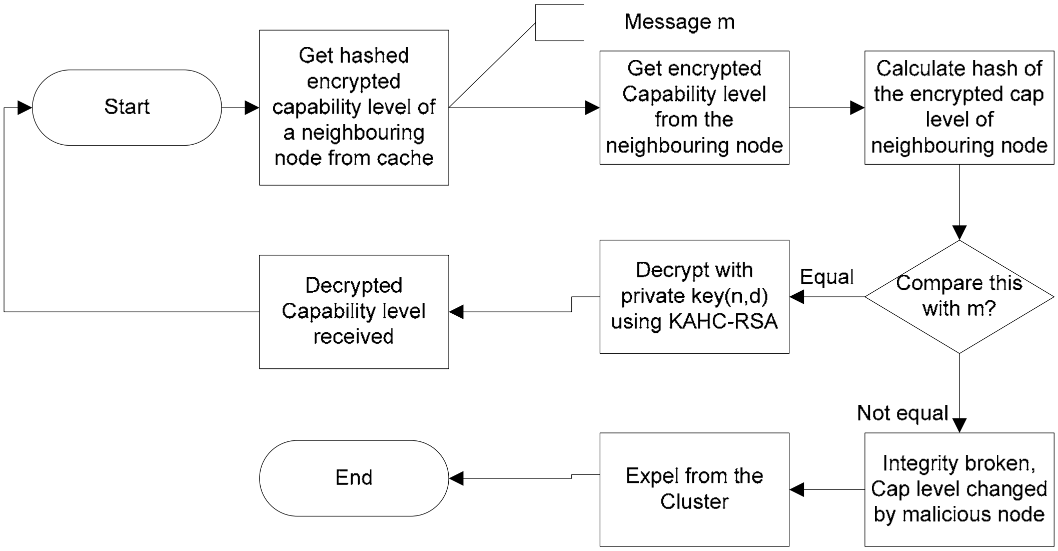

Message m
Start
Get hashed encrypted capability level of a neighbouring node from cache
Get encrypted Capability level from the neighbouring node
Calculate hash of the encrypted cap level of neighbouring node
Compare this with m?
Equal
Decrypt with private key(n,d) using KAHC-RSA
Decrypted Capability level received
Not equal
Integrity broken, Cap level changed by malicious node
Expel from the Cluster
End

Figure 4.3 Decryption using KAHC-RSA

4.5.3.8 Cluster Configuration:

After extracting original capability level of nodes, clusters will be configured. Each node will be assigned a cluster identifier through standard cluster configuration procedure of HCNP.

4.5.4 Factors involved in determining key combinations:

This is very important to find the best key combination for KAHC-RSA. There are many factors that are important in determining key combination suitable for that particular situation. Those factors are:

- Network bandwidth
- Level of security required
- Physical capabilities of nodes i.e. overall trend of nodes physical resources.

If available network bandwidth is sufficient enough to handle higher key size values without compromising much on efficiency, KAHC-RSA key sizes can be adjusted to higher bits key sizes combinations. This will provide higher security to the nodes and the architecture as well. More secure clusters will be configured and applications specifically designed for such architecture, will have to accommodate a bit less security measures in it.

If available network bandwidth is the main issue and fluctuates abruptly at different timings, the smaller key sizes combination can give more suitable results. But, less security will be provided to the architecture and hence to the node. So, key combination is basically a tradeoff between the level of security required and the availability of bandwidth. Higher security will definitely cost more than lower security in terms of time especially.

Another important thing is, the physical capabilities of nodes play very important role in determining the best suitable key combinations. If overall trend of nodes is such that many nodes have very low in sharing their physical resources than no need to implement higher security key combinations and vice versa.

4.5.5 Key Combination Examples:

Some key combinations are used for experimentation. These dynamically assigned key size combinations according to heterogeneous physical resources of nodes are given below:

Keys between 4 and 64:

Number of capability levels is 5, so 5 key sizes are used in this key combination i.e. 4, 8, 16, 32, 64.The higher the node in its capability (1 is the highest and 5 is the lowest level), the higher bits are used to calculate key and vice versa. The Table 4.1, 4.2, 4.3, 4.4 and 4.5 show five different combinations.

Node Id	Cap Level	RSA bits
1	5	4
2	2	32
3	3	16
4	4	8
5	1	64
6	4	8
7	3	16
8	5	4
9	1	64
10	2	32

Table 4.1 Keys between 4 and 64 (combination 1)

Keys between 8 and 128 (8, 16, 32, 64, and 128):

Node Id	Cap Level	RSA bits
1	5	8
2	2	64
3	3	32
4	4	16
5	1	128
6	4	16
7	3	32
8	5	8
9	1	128
10	2	64

Table 4.2 Keys between 8 and 128 (combination 2)

Keys between 16 and 256 (16, 32, 64, 128, and 256):

Node Id	Cap Level	RSA bits
1	5	16
2	2	128
3	3	64
4	4	32
5	1	256
6	4	32
7	3	64
8	5	16
9	1	256
10	2	128

Table 4.3 Keys between 16 and 256 (combination 3)

Keys between 32 and 512 (32, 64, 128, 256 and 512):

Node Id	**Cap Level**	**RSA bits**
1	5	32
2	2	256
3	3	128
4	4	64
5	1	512
6	4	64
7	3	128
8	5	32
9	1	512
10	2	256

Table 4.4 Keys between 32 and 512 (combination 4)

Keys between 64 and 1024 (64, 128, 256, 512 and 1024):

Node Id	**Cap Level**	**RSA bits**
1	5	64
2	2	512
3	3	256
4	4	128
5	1	1024
6	4	128
7	3	256
8	5	64
9	1	1024
10	2	512

Table 4.5 Keys between 64 and 1024 (combination 5)

4.6 Secure-HCNP Data Structure

Each HCNP node maintains the following information in its cache:

1-Node Id 2- Time Stamp 3- Cluster Id 4- Capability level.

Let’s suppose that the cache is comprised of 6 nodes, as shown in Figure 4.4.

1/0/3/4	2/2/5/11	6/1/2/5	8/1/3/2	10/2/5/2	4/1/3/11

Figure 4.4 HCNP Node Cache

The first entry in the cache i.e. ‘1/0/3/4’ contains a node’s information in the following format: Node Id=1, Time Stamp=0 (depicting freshest node, it increases when node remains for the longer time period), Capability Level=3 (can be Integral Capability Level if multiple physical parameters are used), Cluster Id=4.

The Secure-HCNP maintains some extra information in its fields and its cache looks as in Figure 4.5 and Figure 4.6.

1- Node Id, 2- Time Stamp, 3- Encrypted Capability Level, 4- Cluster Id, 5- Security Identifier (1/0).

1/0/**505DGluQ3mZHO0lo/+eoLA**
/4/1

Figure 4.5 Single Entry of Secure HCNP Node's Cache

1/0/3/4/1	6/1/2/5/1	8/1/3/2/1	10/2/5/2/1	4/1/3/11/1

Figure 4.6 Secure HCNP Cache

Secure-HCNP stores the architecture specific information in hashed encrypted format. The other protocol related information is stored in the normal format just as in case of HCNP. The Figure 4.5 shows one entry '1/0/3/4/1' of the cache. Here, '3' shows capability level of a node which is finally stored in the cache in encrypted format. The Figure 4.7 shows the node and Figure 4.8 shows the overall view of an HCNP cache. The public key encryption technique KAHC-RSA is used for the purpose. Each and every node maintains capability related information in encrypted format. For smaller RSA bits, the hashing may not look as useful as in case of larger bits. For example, when 8 bit standard RSA is used, the length of hashed entry in the cache may seem longer as compared to 512 bit standard RSA, which seems quite compressed from original. The examples are given as under:

8 bit Standard RSA:

Original Message: 2

Encrypted Message: 22985

After MD-5 Hashing (The format in which it is stored in the cache): lK3AwFv34fgldEJRh45GtQ

512 bit Standard RSA:

Original Message: 5

Encrypted Message:

3695017693800482660908501266160783931341245922166016241679153271936307 44

20730281815447014133387276256067228049385890669230506112921241921738207910372170310088814207636370840881271917844808922166924406979786015765809852990500341207516386139059160615388765235642225200560559374034535881387392993257834184649296

After MD-5 Hashing (The format in which it is stored in the cache): YXVlNbzkWC0iob2EJfQ4+w

Result: in both above regardless of the encrypted message length, the generated hash function is of equal length for every message. This includes more security to the node because this is not known to any of the node's that which RSA is used.

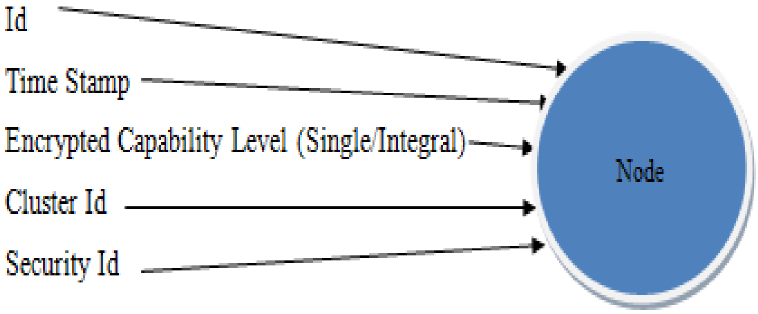


Figure 4.7 Secure-HCNP Node Data Structure

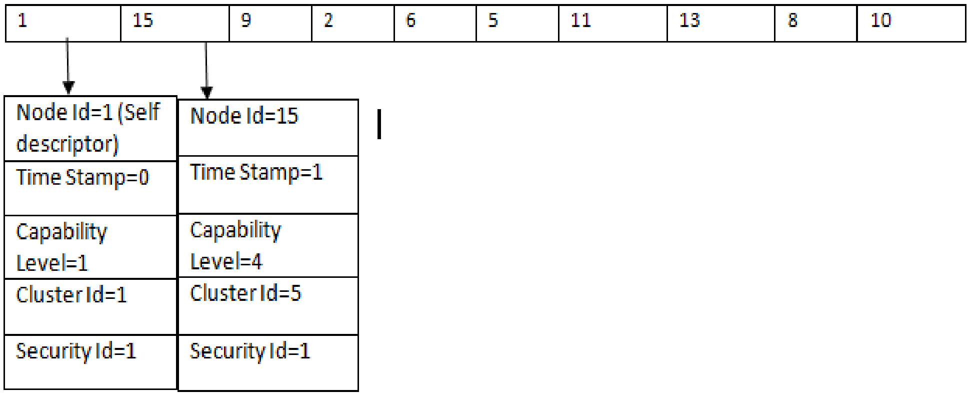


Figure 4.8 Overall View of Secure HCNP Node's Cache

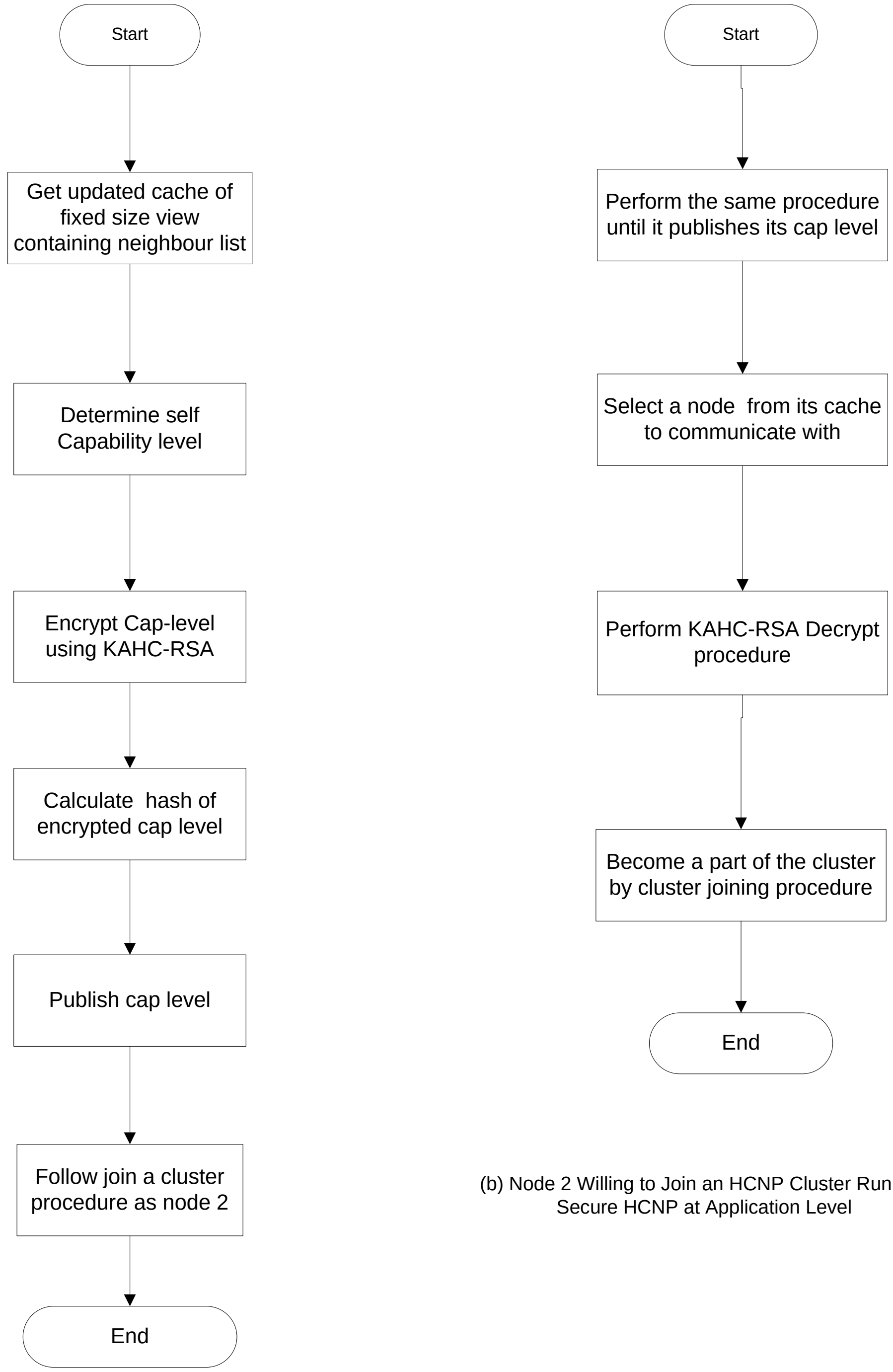
4.7 Secure-HCNP Algorithm

The algorithm showing HCNP procedure is given under:

```
Set TCL value
for  i=1 to cache length
{
    Self:
    Exchange view with neighbors
    Update neighbors list in the cache
    Allocate key size
    Encrypt level with the allocated key size (use any public key)
    Calculate hash using MD-5
    Replace level info in the cache with hashed and encrypted level
    Publish cache (only cap level info)
    Other:
    Extract neighbor node hashed encrypted level
    Calculate hash of the encrypted level
    If (received level values==calculated level value)
    {
        Level info is correct
        Decrypt level info
        Use this info for cluster configuration
        Assign cluster Id to the node
    }
    else
    {
        Set Security Id of neighboring node=0 (depicts the node as malicious)
        Remove the node from the cache
    }
}
```

4.8 Secure-HCNP Procedure

The schematic diagram of Secure-HCNP is drawn for better understanding. Sender and receiver nodes with flow of information, is shown in Figure 4.9 (a) and (b).


Start
Get updated cache of fixed size view containing neighbour list
Determine self Capability level
Encrypt Cap-level using KAHC-RSA
Calculate hash of encrypted cap level
Publish cap level
Follow join a cluster procedure as node 2
End
Start
Perform the same procedure until it publishes its cap level
Select a node from its cache to communicate with
Perform KAHC-RSA Decrypt procedure
Become a part of the cluster by cluster joining procedure
End

(b) Node 2 Willing to Join an HCNP Cluster Running Secure HCNP at Application Level

(a) Node 1 Willing to Join a Secure HCNP Cluster

Figure 4.9 Secure HCNP Procedure

4.9 Secure HCNP Efficiency

Standard RSA with 1024 bits encryption cannot be used in case of HCNP. The major reason is that HCNP is a protocol for designing highly scalable and robust p2p overlay network with support of heterogeneity. If standard RSA is used with it, the network will become extremely slow due to computational complexity of 1024 bit RSA. Also all nodes do not require very high levels of security. To overcome this problem, KAHC-RSA is designed and tested. The Table 4.6 represents some factual results by comparing KAHC-RSA with Standard RSA. It is very clear in the Table 4.6 that at 1024 bits key size Standard RSA needs almost 2.5 sec for encryption and decryption of capability level, whereas, KAHC-RSA requires only 0.5 sec for that. This is at least 5 times faster than standard RSA. The range column of Table 4.6 shows the minimum and maximum key sizes and the key sizes in between those minimum and maximum. This depicts heterogeneous key assignment in KHAC-RSA. The last column of Table 4.6 shows bit size reduction ratio in case of KAHC-RSA. Bit size reduction ratio is the ratio between the maximum key size used in a combination and the optimal key size achieved.

	Optimal Key Size (bits)	**Range = minimum to maximum bits used for KAHC-RSA**	**Standard RSA**	**KAHC-RSA**	**Bit size Reduction Ratio**
Average Total Time(μ sec)	4	4 to 64	248	-	
	8	4 to 64, 8 to 128	450	-	
	16	4 to 64, 8 to 128, 16 to 256	765	-	
	32	4 to 64, 8 to 128,16 to 256,32 to 512	1345	-	
	64	4 to 64, 8 to 128,16 to 256, 32 to 512, 64 to 1024	3085	-	
	128	8 to 128,16 to 256, 32 to 512, 64 to 1024	9736	-	
	256	16 to 256, 32 to 512, 64 to 1024	37100	-	
	512	32 to 512, 64 to 1024	237025	-	
	1024	64 to 1024	**2309810≈** 2.5 sec	-	
	24≈ 25	4 to 64 (Comb 1)	-	977	2.6
	48≈50	8 to 128 (Comb 2)	-	3009	2.6
	98≈100	16 to 256 (Comb 3)	-	10269	2.6
	194≈200	32 to 512 (Comb 4)	-	55595	2.6
	390≈400	64 to 1024 (Comb 5)	-	**498029≈** 0.5 sec	**2.6**

Table.4.6 Comparison of KAHC-RSA with original RSA

Bit size for each key combination is reduced to 2.6th of the maximum key size used. So, KAHC-RSA is computationally faster from original RSA by ratio of 2.6.

CHAPTER 5: SIMULATIONS & RESULTS

5.1 PeerSim Overview

PeerSim [4] is an open source P2P systems simulator developed at the Department of Computer Science, University of Bologna, Italy. It has the following features:

- It has been developed with Java
- Available on Source Forge *(peersim.sf.net)*
- Its aim is to cope up with P2P systems
- It is highly Scalable (up to 1 million nodes) and Highly configurable

The experiments are done on the above P2P simulation platform, PeerSim, to see the effect of all changes done. PeerSim has been developed to cope with the high dynamicity of the systems (nodes), and thus to accomplish extreme scalability.

PEERSIM is an extremely scalable simulation environment that supports dynamic scenarios such as churn and other failure models. Protocols are needed to be specifically implemented for the PEERSIM Java API. However, with a reasonable effort they can be evolved into a real implementation. Testing in specified parameter-spaces is supported as well.

PeerSim [4] engine supports two types of simulations:

a. Cycle driven (CD Simulator)

b. Event driven (ED Simulator)

In cycle driven simulator, the concept of cycles or rounds is used. One round or cycle means one complete run of the set of nodes. A new cycle starts when all the nodes are done with the experiment. In Event driven simulator, experiment is triggered by an event or message generated by any of the network nodes.

A configuration file is required to provide the Simulator with network configuration parameters such as network size, base protocols used etc. An example of the configuration file is given as under:

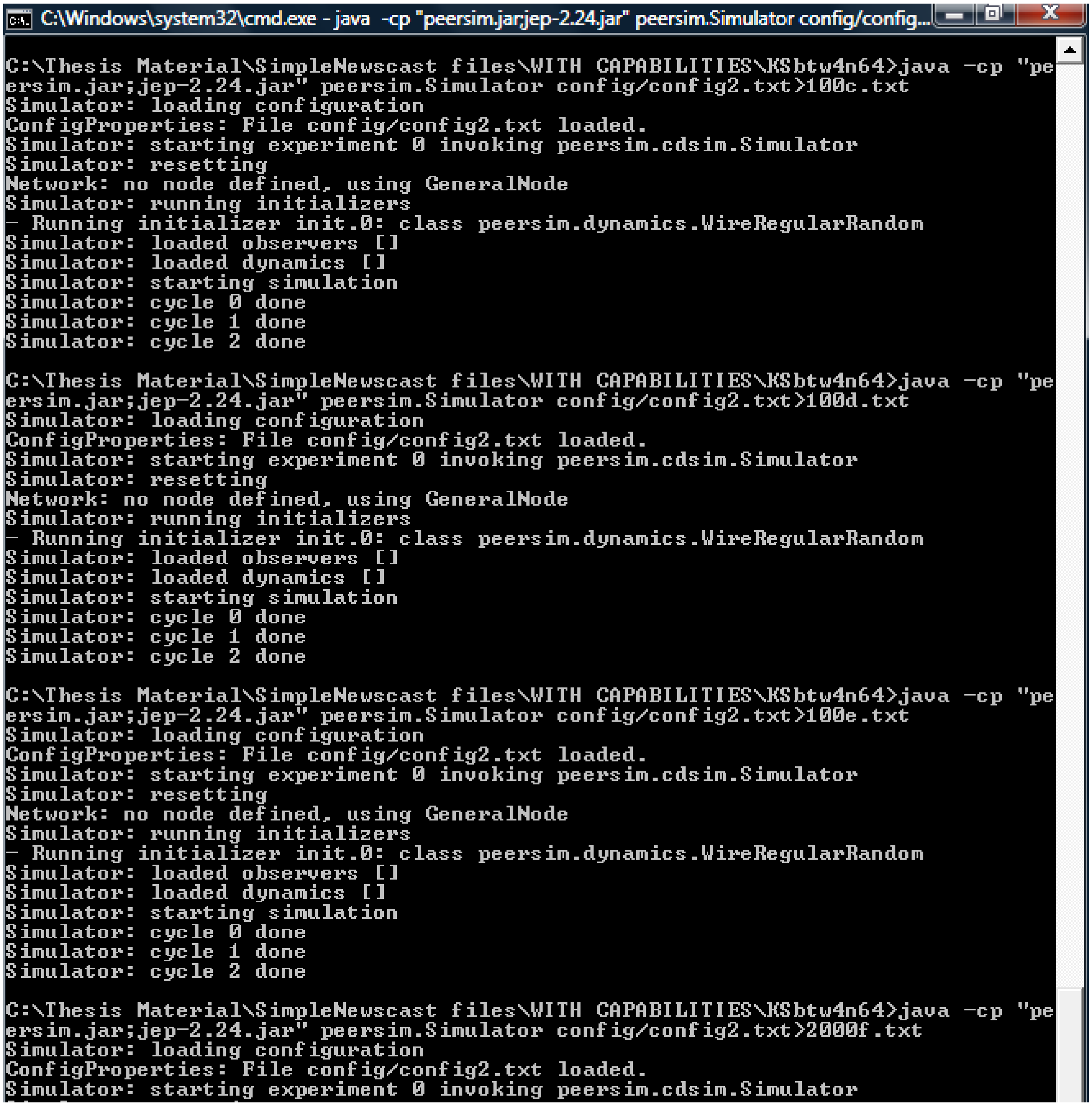
```
# PEERSIM EXAMPLE 1
random.seed 1234567890
simulation.cycles 3
```

```
overlay.size 100
overlay.maxsize 100
protocol.0 example.SimpleNewscast.SimpleNewscast
protocol.0.cache 40
init.0 peersim.dynamics.WireRegularRandom
init.0.protocol 0
init.0.degree 40
```

The Secure-HCNP is tested and assessed for its efficiency and dynamicity on PeerSim (a p2p simulation test bed). Effects of change in different configuration parameters, is also studied.

5.2 Simulations

The snapshot of running simulations on PeerSim is shown in Figure 5.1.


C:\Windows\system32\cmd.exe - java -cp "peersim.jar;jep-2.24.jar" peersim.Simulator config/config...

```
C:\Thesis Material\SimpleNewscast files\WITH CAPABILITIES\KSbtw4n64>java -cp "pe
ersim.jar;jep-2.24.jar" peersim.Simulator config/config2.txt>100c.txt
Simulator: loading configuration
ConfigProperties: File config/config2.txt loaded.
Simulator: starting experiment 0 invoking peersim.cdsim.Simulator
Simulator: resetting
Network: no node defined, using GeneralNode
Simulator: running initializers
- Running initializer init.0: class peersim.dynamics.WireRegularRandom
Simulator: loaded observers []
Simulator: loaded dynamics []
Simulator: starting simulation
Simulator: cycle 0 done
Simulator: cycle 1 done
Simulator: cycle 2 done

C:\Thesis Material\SimpleNewscast files\WITH CAPABILITIES\KSbtw4n64>java -cp "pe
ersim.jar;jep-2.24.jar" peersim.Simulator config/config2.txt>100d.txt
Simulator: loading configuration
ConfigProperties: File config/config2.txt loaded.
Simulator: starting experiment 0 invoking peersim.cdsim.Simulator
Simulator: resetting
Network: no node defined, using GeneralNode
Simulator: running initializers
- Running initializer init.0: class peersim.dynamics.WireRegularRandom
Simulator: loaded observers []
Simulator: loaded dynamics []
Simulator: starting simulation
Simulator: cycle 0 done
Simulator: cycle 1 done
Simulator: cycle 2 done

C:\Thesis Material\SimpleNewscast files\WITH CAPABILITIES\KSbtw4n64>java -cp "pe
ersim.jar;jep-2.24.jar" peersim.Simulator config/config2.txt>100e.txt
Simulator: loading configuration
ConfigProperties: File config/config2.txt loaded.
Simulator: starting experiment 0 invoking peersim.cdsim.Simulator
Simulator: resetting
Network: no node defined, using GeneralNode
Simulator: running initializers
- Running initializer init.0: class peersim.dynamics.WireRegularRandom
Simulator: loaded observers []
Simulator: loaded dynamics []
Simulator: starting simulation
Simulator: cycle 0 done
Simulator: cycle 1 done
Simulator: cycle 2 done

C:\Thesis Material\SimpleNewscast files\WITH CAPABILITIES\KSbtw4n64>java -cp "pe
ersim.jar;jep-2.24.jar" peersim.Simulator config/config2.txt>2000f.txt
Simulator: loading configuration
ConfigProperties: File config/config2.txt loaded.
Simulator: starting experiment 0 invoking peersim.cdsim.Simulator
```

Figure 5.1 Running Simulations on PeerSim

5.3 Experiments and Results

Different experiments are done to study Secure HCNP characteristics. All the experiments used 3 cycles and initially 500 nodes overlay. The cache and the degree size used is 40.

The following experiments are done using PeerSim:

1-Dynamic assignment of key sizes

The Figure 5.2 (a) and (b) shows dynamic assignment of key sizes. Key sizes range from 4 to 1024 bits and five different key combinations are used i.e. 4 to 64 bits, 8 to 128bits , 16 to 256 bits , 32 to 512 bits and 64 to 1024 bits.

The CL =5. There are five differnet key combinations are used in five different runs. The slight decrease in key sizes can be seen in Figure 5.2 (a). For CL=1, 5 combinations are tried ranging from 64 bit to 1024 bits. Similarly for CL=2, combinations are from 32 to 512 and so on. With the increase in physical capability of the node, the key size assigned increases. The Lower value of level depicts the node is higher in its physical capability, e.g. CL=1 means the node with highest resources and CL=5 means the node with lowest physical capability. This can be reversed dependingon the choice of secure-HCNP users. This distribution is uniform across all clusters configured.

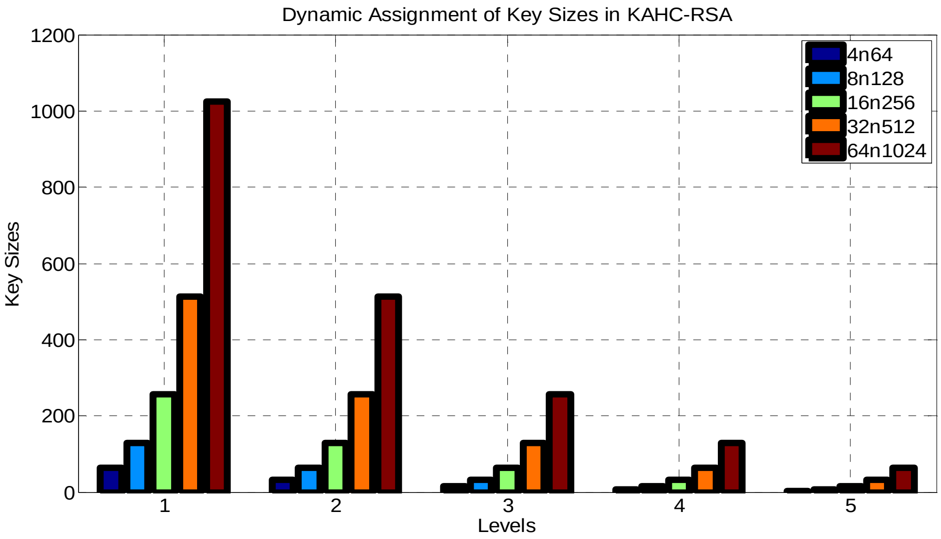


Figure 5.2 (a) Dynamic Assignment of Key Sizes in KAHC-RSA

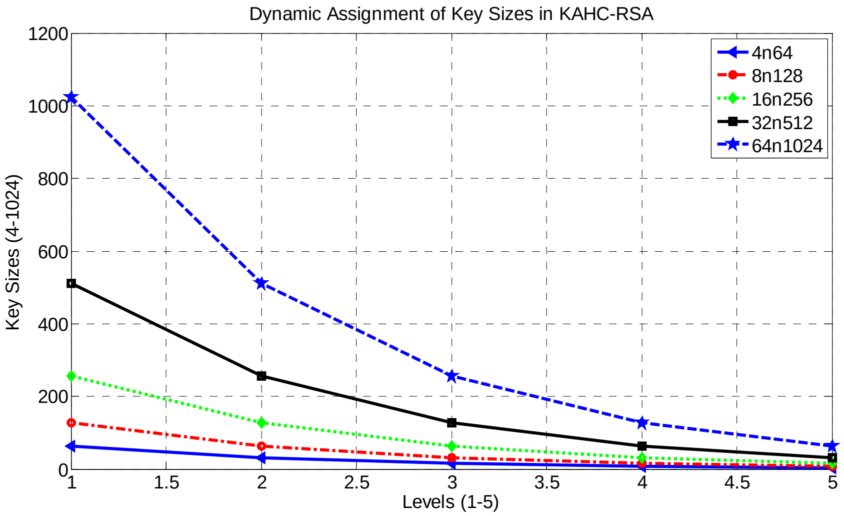


Figure 5.2 (b) Dynamic Assignment of Key Sizes in KAHC-RSA

2- Key size distribution across a cluster

The Figure 5.3 shows the assignment of correct key sizes to randomly selected 20 nodes. Other parameters are the same for this experiment as of the above.

The node with node Id=1 is assigned 4 bit key and the node 19 is assigned 64 bit key. Key sizes range for this experiment is 4 to 64 bits. The important assessment is the heterogeneity of key sizes is totally according to heterogeneity of nodes capabilities. So, basic property of HCNP is preserved.

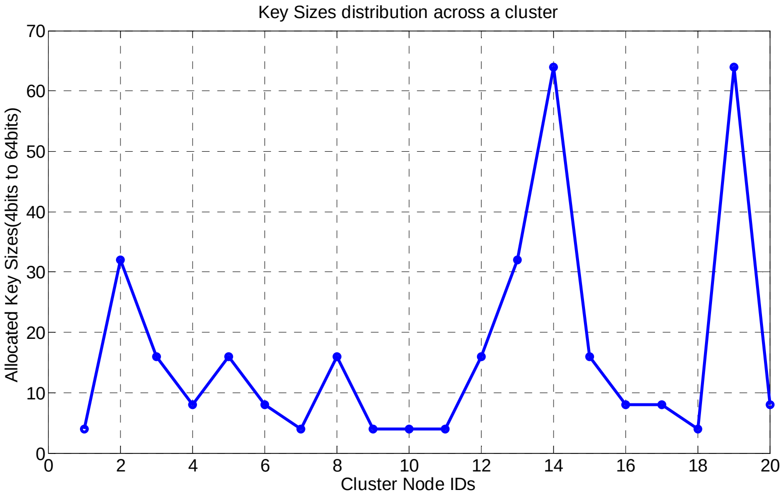


Figure 5.3 Key Size Distribution across a Cluster

3- Level distribution across a cluster

Similarly, for the range 8 bit to 128 bits, the same results are drawn. Level distribution is also studied across a cluster of 40 nodes. The 20 nodes are shown in Figure 5.4 (a) and Figure 5.4 (b). This also presents heterogeneity applied to nodes. As CL = 1 to 5, so each node is assigned with 5 distinct values depicting its physical capability. It can be noticed that the level is shown just by a digit and any node acting as a malicious node, cannot guess the exact value of nodes physical capability.

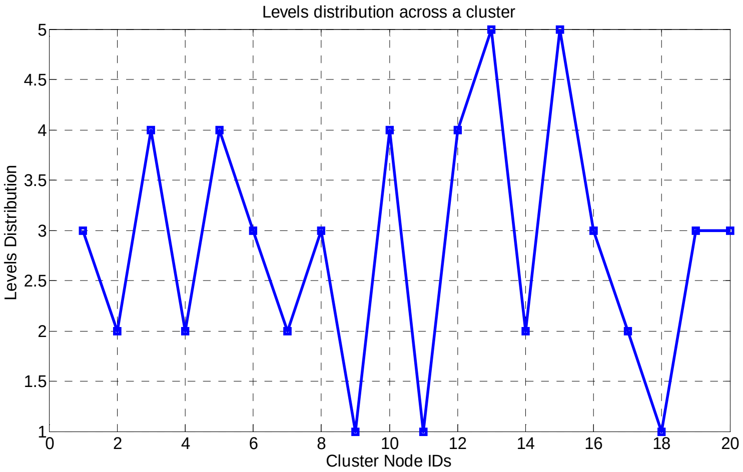


Figure 5.4 (a) Level Distribution across a Cluster

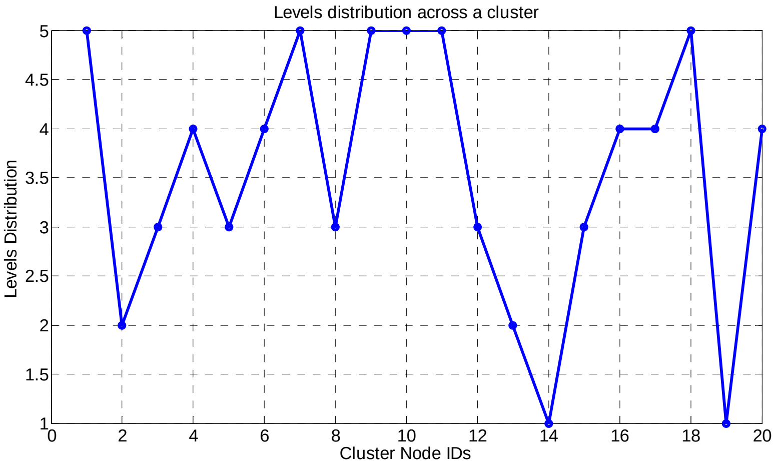


Figure 5.4 (b) Level Distribution across a Cluster

4- Average Encryption, decryption and hashing time of KAHC-RSA

The Figure 5.5 (a) shows the comparison of average encryption, decryption and hashing time for KAHC-RSA. The decryption process takes the longest in KAHC-RSA because of involvement of modulus calculation time. It is necessary to get the private key.This is observed for network size=1000 to 10000. Cluster sizes vary from 40 nodes to 200 nodes per cluster.

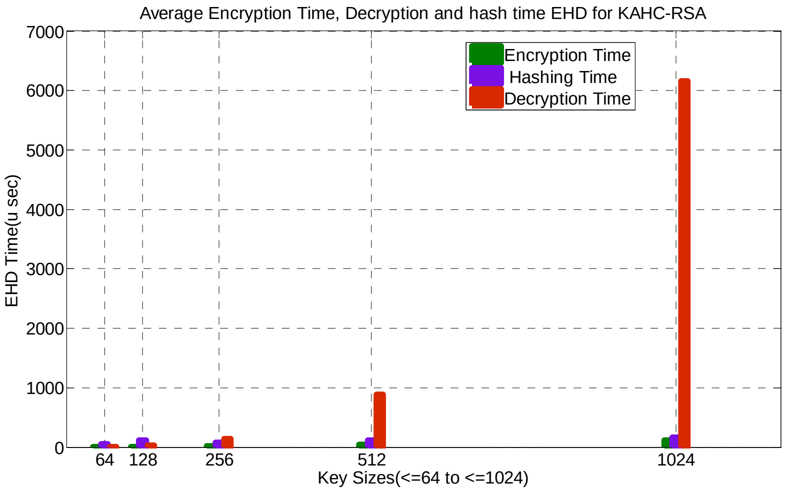


Figure 5.5 (a) Average Encryption, Decryption and Hashing Time of KAHC-RSA

5- Average Encryption, decryption and hashing time of Standard RSA

The Figure 5.5 (b) shows the same as above for Standard RSA.

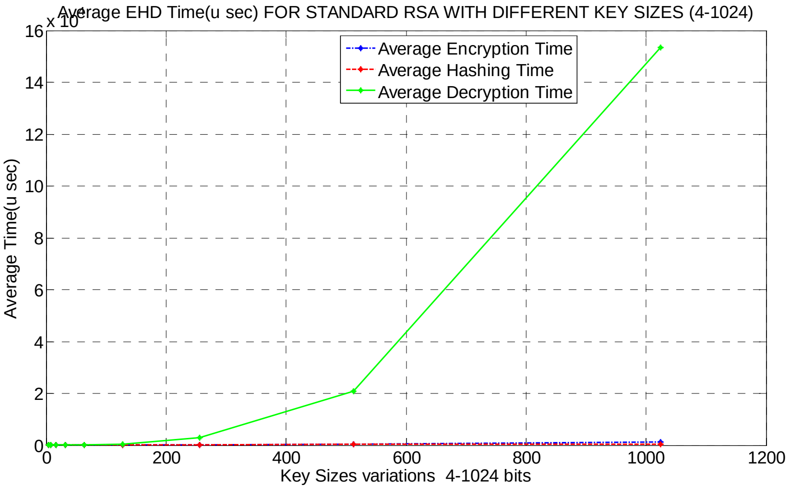


Figure 5.5 (b) Average Encryption, Decryption and Hashing Time of Standard-RSA

The Figure 5.6 shows that the total time is composed of maximum of modulus calculation time in KAHC-RSA as well as original RSA. Encryption and decryption time is negligible as compared to modulus calculation time.

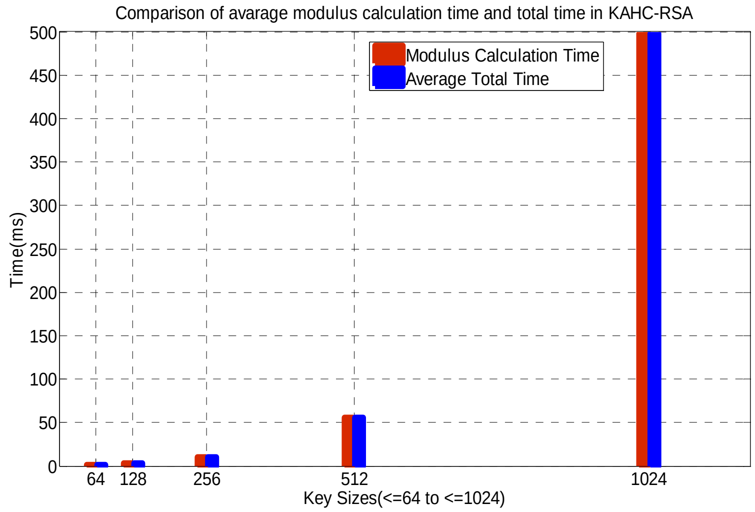


Figure 5.6 Comparison of Modulus Calculation Time with Total Time in KAHC-RSA

6- Average efficiency of Standard RSA

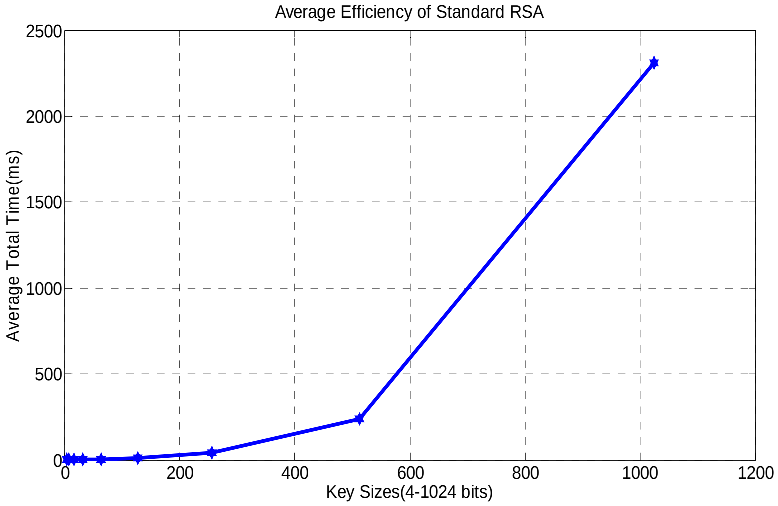


Figure 5.7 (a) Average Efficiency of Original RSA

The Figure 5.7 (a) demonstrates the efficiency of Standard RSA in terms of total time required to encrypt and decrypt the capability levels. It is found that there is an exponential rise in time with increase in key sizes. The key sizes taken for experimentation are 4, 8, 16, 32, 64, 128, 256, 52 and 1024 bits. The maximum time for 1024 bit (as in case of standard RSA), is 2500 ms, Whereas, the maximum time for 64 to 1024 bit KAHC-RSA is, 500 ms as shown in Figure 5.7 (b). So, KAHC-RSA is proven to be five times faster than standard RSA, And more suitable for HCNP because of its compatibility with HCNP property i.e. heterogeneity.

7- Average efficiency of KAHC-RSA

In Figure 5.7 (b), average efficiency of KAHC-RSA is shown. It is quite clear that there is an exponential rise in average total time with increase in key sizes. The difference from standard RSA is that the maximum time for 1024 bit standard RSA is 2500 ms whereas in case of KAHC-RSA, it is 500ms which proves KAHC-RSA to be five times faster than original RSA. Figure 5.7 (c) shows the same fact.

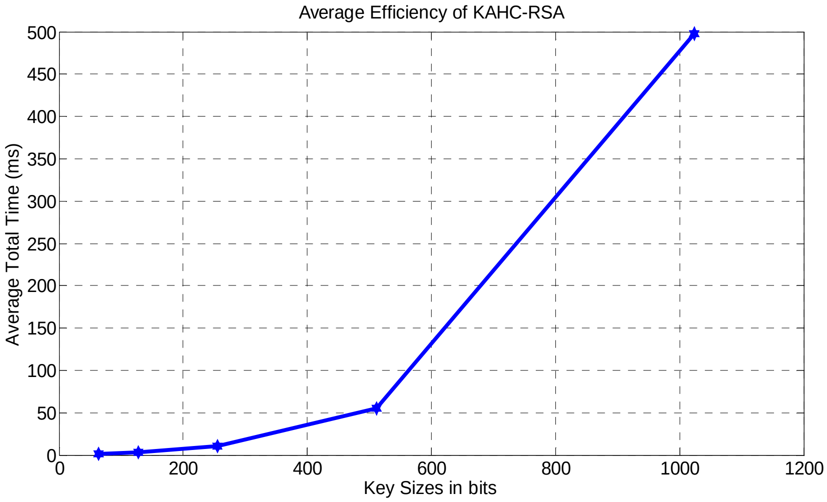


Figure 5.7 (b) Average Efficiency of KAHC- RSA

8- Comparison of average efficiency of original vs KAHC-RSA

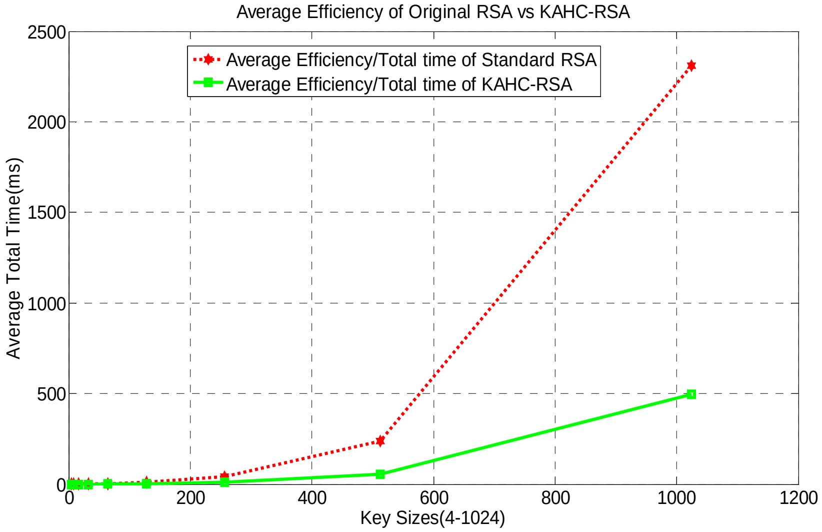


Figure 5.7 (c) Comparison of Average Efficiency of Original RSA with KAHC-RSA

9- Cluster wise average total time in KAHC-RSA

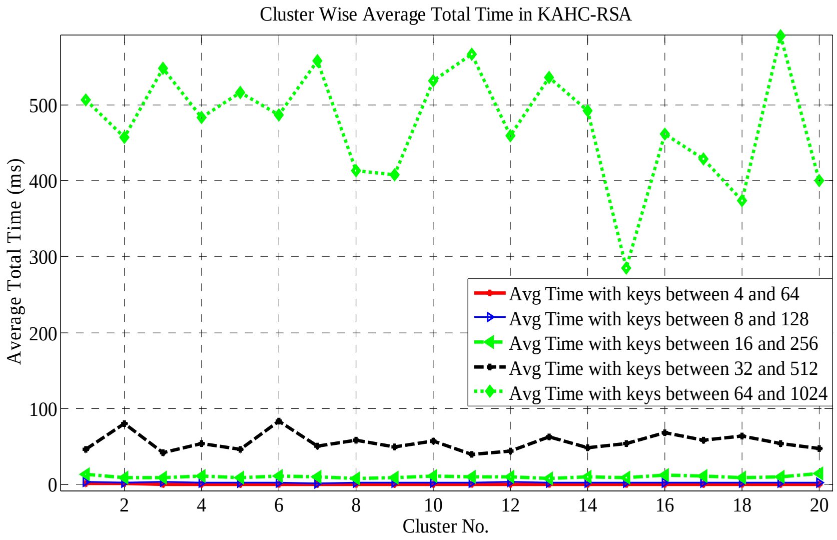


Figure 5.8 Cluster-Wise Average Total Time in KAHC- RSA

In Figure 5.8, cluster wise performance of KAHC-RSA is observed. There are total of 20 clusters, each composed of nearly 40 nodes. The average total time for 64 to 1024 bits KAHC-RSA is fluctuating around 50ms as shown in the Figure 5.8. For all other categories, KAHC-RSA performance is below 100ms. This means regardless of nodes capabilities, the average total time is almost the same in all the 20 clusters for one range of key size values. The clusters are picked randomly from the overlay.

10- Cluster wise average key sizes in KAHC-RSA

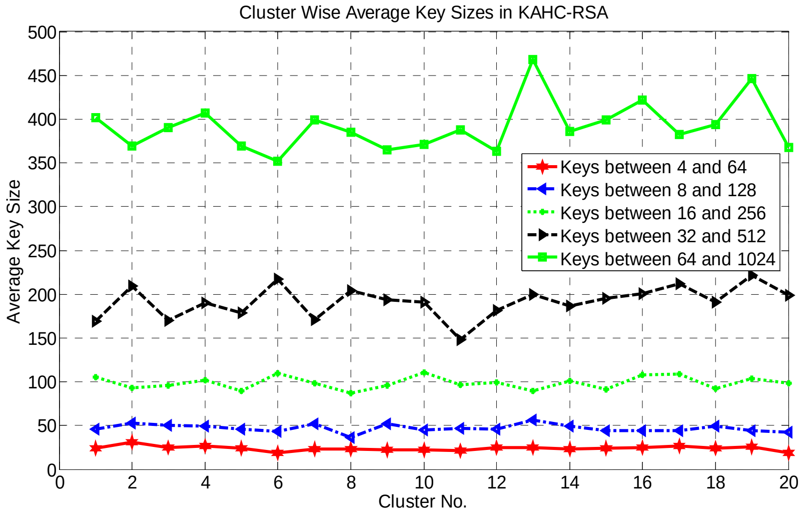


Figure 5.9 Cluster-Wise Average Key Sizes in KAHC- RSA

Similarly, the average key size is studied in the Figure 5.9 in 20 clusters. Main axis shows clusters and y axis shows average key sizes for five different ranges of key sizes in KAHC-RSA. These factual values are found very close to the values obtained theoretically. The values for individual clusters match the average of total, say 500 clusters. This shows optimal key value for a cluster can be taken for uniform security implementation.

11-Optimal Key Size in KAHC-RSA

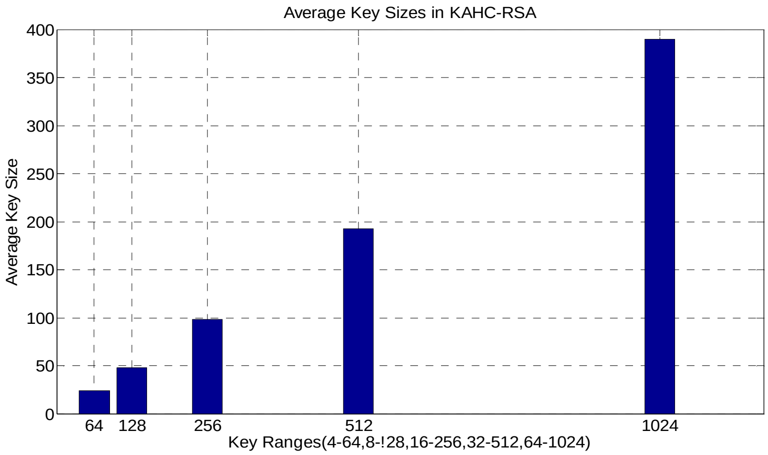


Figure 5.10 Optimal Key Size in KAHC- RSA

The Figure 5.10 shows the average key sizes i.e. optimal key size values for five different key size ranges (key combinations) in KAHC-RSA. The CL=5, 20,000 nodes, more than 500 clusters, overlay with degree=40 and cluster size=40. This optimal key size can be used for uniform key size implementations. Optimal key size will provide security somewhat between maximum and minimum key sizes. For example, instead of using 64 to 1024 key size ranges, the optimal key size i.e. 390 bit value can be used.

12- Effect of network size on efficiency of KAHC-RSA

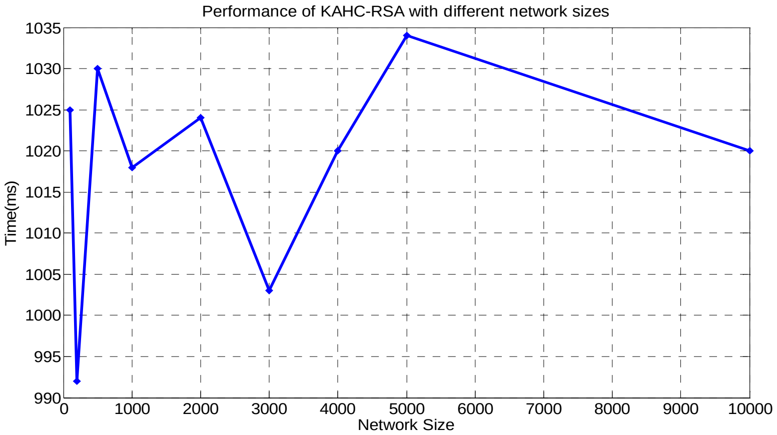


Figure 5.11(a) Effect of Network Size on Efficiency of KAHC- RSA

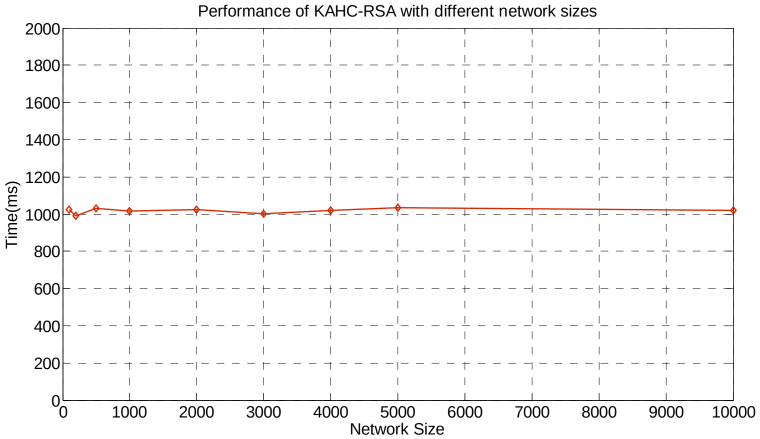


Figure 5.11 (b) Effect of Network Size on Efficiency of KAHC- RSA

The effect of network size on the performance of KAHC-RSA is shown in Figure 5.11 (a) and Figure 5.11 (b). The experiment takes 1000, 2000, 3000, 4000, 5000 and up to 10,000 nodes with five different key ranges. One of the key size ranges i.e. 32 to 512 is shown in Figure 5.11(a) and Figure 5.11(b). The average time remains the same for all network sizes, means that increase in network size does not have any impact on the performance of KAHC-RSA. The average total time remains the same for 1000 as well as for 10,000 nodes overlay. Figure 5.11 (a) shows enlarged view of y-scale in Figure 5.11 (b).

13- Comparison of Original RSA on HCNP and High Speed RSA

Figure 5.12 demonstrates the results of comparison between Standard RSA when implemented with HCNP and a high speed implementation of RSA by RSA laboratories

[31]. In Figure 5.12, average total time (required for encryption + decryption) in milliseconds, is drawn against the key sizes (in bits) for both original RSA as well as a High Speed RSA implementation. The Figure 5.12 shows that for key sizes 128 bits and 256 bits, there is not a big difference in total time required for encryption and decryption in both cases whereas with key sizes 512 bits and 1024 bits, the average total time increases exponentially. This shows that the average increasing trend is found in both implementations of RSA when key size is increased. Secondly, when both of them are compared with each other, original-RSA is proven to be more efficient in terms of total time required in encryption and decryption when implemented on HCNP. Originally, the standard RSA implementation is supposed to be lesser efficient than the high speed implementation of RSA if HCNP is not used as an underlying topology.

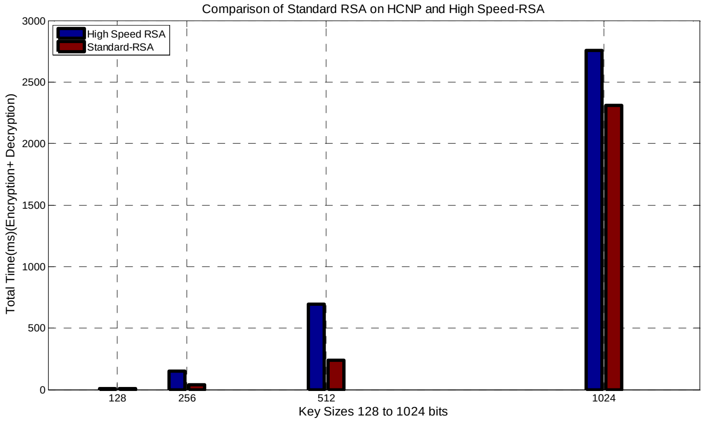


Figure 5.12 Comparison of Original RSA on HCNP and High Speed RSA

14- Comparison of KAHC-RSA and High Speed RSA

When KAHC-RSA is compared with a high speed implementation of RSA, it is found that KAHC-RSA is much faster than the high speed RSA in terms of total time required for encryption and decryption. The Figure 5.13 shows that with increase in key size from 128 bits to 1024 bits, total time also increases. But this increasing trend is more significant in case of higher key size values. Secondly, KAHC-RSA is proven to be much more efficient than a high speed RSA implementation in terms of computational time required to encrypt and decrypt a fixed length message.

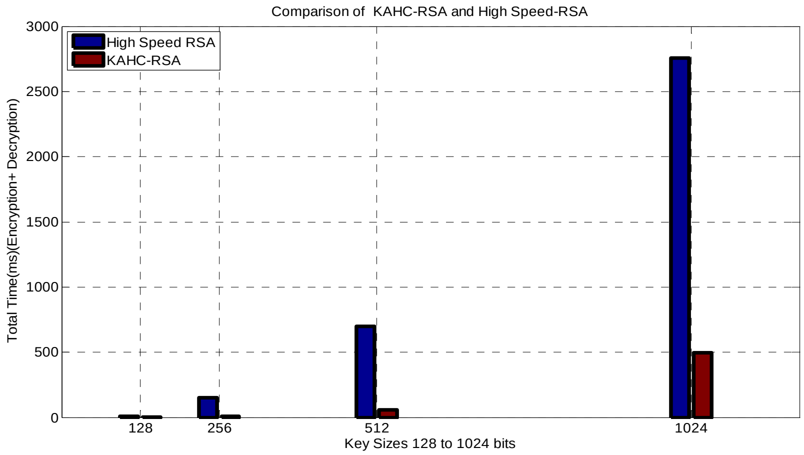


Figure 5.13 Comparisons of KAHC-RSA and High Speed RSA

This also shows that the selection of encryption or decryption algorithm depends on the usage scenario and level of security required because 1024 bit RSA or KAHC-RSA gives higher security than 128 bit but efficiency will be compromised in the former case.

15-Comparing Computational Complexity of KAHC-RSA with original RSA and CRT RSA

When 1024 bit original RSA is compared with one of its fast variants CRT RSA [6] designed for mobile phones (based on Chinese Remainder Theorem) and KAHC RSA, it is found that KAHC-RSA is much faster than original RSA. The column showing decryption time in table 5.1, demonstrates average decryption time in ms for all the three variants of RSA. The decryption time for KAHC-RSA is 498 ms which is less than the decryption time of CRT RSA (especially design to work on mobile phones) i.e. 558 milliseconds.

Similarly, the results presented under the column heading ‘Computational Complexity’ in Table 5.1, show that KAHC-RSA is reduced in its complexity by the factor of 2.5 when compared with original RSA. KAHC-RSA requires lesser computation than CRT RSA and so emerged as a faster variant of RSA than CRT RSA.

	Encryption Time	**Decryption Time (ms)**	**Computational Complexity**
Original RSA	29	2098	$O(K^3)$
CRT RSA	-	558	$2.O((K/2)^3)$
KAHC-RSA	-	498	$(K/2.5)^3$

Table 5.1 Comparison of Decryption Time

Here, K is the number of bits used for key calculation. Figure 5.14 demonstrates the results of Table 5.1 in graphical form. Here, for key size of 512 bits, only Standard RSA and KAHC-RSA is drawn. There are no experiments found in the related research on CRT RSA for 512 bits key size.

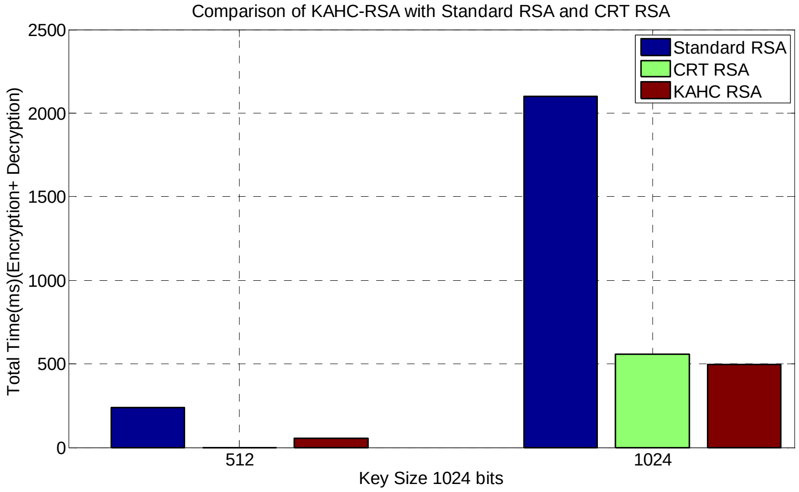


Figure 5.14 Comparison of Decryption Time

CHAPTER 6: CONCLUSION & FUTURE WORK

6.1 Conclusion

The concept of heterogeneous clusters given by HCNP and HCNP with multiple physical parameters can be used and implemented with any gossip-based protocols working on the basis of Peer Sampling Services e.g. Newscast used with HCNP and HCNP with multiple physical parameters. Whereas, the attack model described in previous relevant researches on gossip-based protocols security, can completely destroy the basic properties of any gossip-based protocol. To stop the spread of a poisonous cache and to prevent HCNP from other types of worms and threats, encryption is a good idea. It is concluded that Secure-HCNP can be used to prevent from the attacks described. It uses KAHC-RSA which is a modified version of Original RSA and specifically designed for HCNP. It not only preserves the basic property of HCNP (i.e. heterogeneity) and the concept of heterogeneous security levels and dynamic key assignment, makes it more appropriate in the situations where smaller pieces of information are required to be encrypted. KAHC-RSA can also be used with other gossip based protocols using Peer Sampling Services. The same is experimented with the Newscast protocol in the current research without compromising on efficiency. KAHC-RSA uses java which is proven to be the fastest language for implementation of RSA. KAHC-RSA is proven to be five times faster than original RSA and a bit more efficient than CRT which is a faster variant of RSA. Hence, KAHC-RSA joined the group of faster variants of RSA just as it is modified to use on mobile phones. KAHC-RSA can also be used with Cooperative Heterogeneous Clusters without compromising on efficiency. The experiments done for testing KAHC-RSA for efficiency have shown that KAHC-RSA can equally be used for encrypting any smaller piece of information which is not specific to peer to peer network protocol.

6.2 Limitations

The current research only deals in architecture-related security, not the application-level security. It makes the overlay architecture secure, not the application related data exchanged between them. It is assumed that the application using this protocol will have embedded security specific to the application. Moreover, only smaller pieces of information (e.g. physical capability level or id type information) can be encrypted using KAHC-RSA. Otherwise, the key size may become a bottleneck.

Similarly, in Secure-HCNP only architecture-related information is encrypted using KAHC-RSA. The whole cache possessed by each node is not encrypted. The current research does not handle cache attacks. HCNP cache attacks can also cause inaccessibility to the correct information e.g. if the node id is changed by malicious node or time stamp is modified to some incorrect value. These are basically Newscast related security issues. This problem may disappear if a secure Newscast protocol is used for HCNP.

6.3 Recommendations for Future Work

HCNP is a new protocol and has a lot of room for future work regarding security implementations. Secure HCNP is the first step towards making this protocol secure. In the current research, only Standard/ original RSA is implemented and tested with modification. Other faster variants of RSA (e.g. CRT, R-Prime, Multi power, Multi prime and Re-balanced RSA) should be implemented and tested with java to give better and more efficient results on HCNP. Moreover, other public key cryptographic techniques can be used with HCNP and the efficiency may be calculated. Node authentication can be implemented with more sophisticated methods as compared to the general method used for gossip-based protocols. Similarly, the whole cache encryption techniques may be introduced with faster and more secure implementations.

REFERENCES

[1] Shen Guicheng, Liu Bingwu and Zheng Xuefeng, U. (1971). "Research on Fast Implementation of RSA with Java", International Symposium on Web Information Systems and Applications, pp. 186 – 189, WISA 2009.

[2] Irum Kazmi, Saira Aslam, M. Y. Javed, "Cluster-based Peers Configuration Using HCNP in Peer-to-Peer Overlay Networks", CICSyn 2010.

[3] Irum Kazmi, Saira Aslam, M. Y. Javed, "Cluster-based Peers Configuration with Multiple Physical Parameters using HCNP in Peer-to-Peer Overlay Networks", ICCAIE 2010.

[4] A. Montresor, M. Jelasity, "Peersim: Scalable P2P Simulator", NAPA-WINE Project, 2009.

[5] C. L. Wu, D. C. Lou, and T. Chang, "Computational Complexity Analyses of Modular Arithmetic for RSA Cryptosystem", The 23rd workshop on Combinatorial Mathematics and computation theory, 2006.

[6] K. Hansen, T. Larsen and K. Olsen, "On the Efficiency of Fast RSA Variants in Modern Mobile Phones", International Journal of Computer Science and Information Security, Vol. 6, Issue No. 3, 2009

[7] Jason Hinek and David R.Cheriton, "On the security of multi-prime RSA", Journal of Mathematical Cryptography, Volume 2, Issue 2, pp 117-147, 6-7-2008

[8] V. Vishnumurthy and P. Francis, "A Comparison of Structured and Unstructured P2P Approaches to Heterogeneous Random Peer Selection", USENIX Annual Technical Conference, 2007.

[9] C. Reddy, D. Leonard, and D. Loguinov, "Optimizing Capacity-Heterogeneous Unstructured P2P Networks for Random-Walk Traffic", International Conference on Peer to Peer Computing, IEEE ,2009.

[10] X. Xiao, Y. Shi, Z. Chen and B. Zhang, "On Constructing High Performance Overlay for Layered Streaming in Heterogeneous Networks," Advanced Information Networking and Applications Workshops, pp. 578-584, 2008.

[11] Vishnumurthy and Francis, "On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Overlay Networks", International Conference on

Computer Communications, IEEE, pp. 1-12, 2006.

[12] G. P. Jesi, A. Montresor and M. V. Steen, “ Secure Peer Sampling”, Journal of Computer Networks, Vol. 54,Issue 12, pp. 2086-2098, 2010.

[13] G. P. Jesi and A. Montresor, "Secure Peer Sampling Service: The Mosquito Attack", IEEE International Workshops on Enabling Technologies, pp. 134-139, 2009.

[14] M. Jelasity, A. Montresor, O. Babaoglu, “Detection and Removal of Malicious Peers in Gossip-Based Protocols”, FuDiCo, 2004.

[15] D. C. Lou and C. L. Wu, “Parallel Exponentiation Using Common- Multiplicand-Multiplication and Signed- Digit-Folding Techniques”, International Journal of Computer Mathematics, vol. 81, no. 10, 1187-1202, 2004.

[16] J.C. Ha and S. J. Moon, “A Common-Multiplicand Method to the Montgomery Algorithm for Speeding up Exponentiation,”, Information Proceeding Letters, vol. 66, 105-107, 1998.

[17] L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa, and S. Chien, “A First Look at Peer-to-Peer Worms: Threats and Defenses”, International Symposium on Peer to Peer Systems, MIT, University of Cambridge, 2005.

[18] M. Jelasity, W. Kowalczyk and M. V. Steen, “Newscast Computing”, Technical Report, University of Amsterdam, 2003.

[19] PeerSim, A peer-to-peer simulator by sourceforge.net accessed on 23-06-2010.

[20] Spyros Voulgaris, M´ark Jelasity, M. V. Steen, “A Robust and Scalable Peer-to-Peer Gossiping Protocol”, International workshop on agents and peer-to-peer computing, vol. 2872, pp. 47-58, 2004.

[21] R. Subramanin, B. D. Goodman, “Peer to Peer computing: The evolution of a Disruptive Technology”, Idea Group Publishing, 2005.

[22] A. Bakker and M. V. Steen, “Puppet Cast: A Secure Peer Sampling Protocol”, European Conference on Computer Network Defense, 2008

[23] G. P. Jesi, D. Gavida, C. Gamage, and M. van Steen. “A Secure Peer Sampling Service as “Hub Attack” countermeasure”. Technical Report UBLCS-2006-17, Dept. of Computer Science, University of Bologna, Italy, May 2006.

[24] G. P. Jesi, D. Hales, and M. van Steen. “Identifying Malicious Peers Before It’s Too Late: A Decentralized Secure Peer Sampling Service”. International Conference on Self-Adaptive and Self-Organizing Systems IEEE, June 2007.

[25] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: “Byzantine Resilient Random Membership Sampling”, ACM Symposium on Principles of Distributed Computing , July 2008.

[26] G. P. Jesi, A. Montresor and M. V. Steen, “Secure Peer Sampling”, Journal of Computer Networks, Vol. 54, pp. 2086-2098, 2010 .

[27] Saira Aslam, “Cooperative Heterogeneous Clusters”, MS Final Thesis, Department of Computer Engineering, NUST,2010

[28] C. Wang and Bo Li, “Peer to Peer Overlay Networks: A Survey”, April 20, 2003.

[29] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and Steven Lim, “A Survey and Comparison of Peer to Peer Overlay Network Schemes”, IEEE Communications Survey and Tutorials, March 2004.

[30] Jorge Ardenghi and Javier Echaiz, “Peer-to-Peer Systems: The Present and the Future”, Journal of Computer Science & Technology, Vol. 7, Issue No. 3, October 2007.

[31] C. Kaya Koc, “High Speed RSA Implementation”, Technical Report by RSA Laboratories, RSA Data Security Inc, CA, 2005.

[32] J.Saigeetha and V.Selvi, “Speed and Security Enhancement through Public Key Cryptography”, Journal of Engineering Science and Technology, Vol. 2, Issue no.8, pp. 3551-3556, 2010.

[33] Challa Narasimham and Jayaram Pradhan ,“Evaluation of Performance Characteristics of Cryptosystem using Text Files”, Journal of Theoretical and Applied Information Technology,Vol. 4, Issue. 1 ,pp. 56-60, JATIT 2008 .

[34] R. Biswas, S. Bandyopadhyay and A. Banerjee “A Fast Implementation of The RSA Algorithm Using GNU MP Liberary”, Naional Workshop on Cryptography, 2003.

[35] S. Subasree and N. K. Sakthivel, “Design of a New Security Protocol using Hybrid Cryptography Algorithms.”, International Journal of Research and Reviews in Applied Sciences, Vol. 2, Issue. 2 , IJRRAS 2010.