International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:6, 2016

A Distributed Cryptographically Generated Address Computing Algorithm for Secure Neighbor Discovery Protocol in IPv6

M. Moslehpour, S. Khorsandi

***Abstract*—**Due to shortage in IPv4 addresses, transition to IPv6 has gained significant momentum in recent years. Like Address Resolution Protocol (ARP) in IPv4, Neighbor Discovery Protocol (NDP) provides some functions like address resolution in IPv6. Besides functionality of NDP, it is vulnerable to some attacks. To mitigate these attacks, Internet Protocol Security (IPsec) was introduced, but it was not efficient due to its limitation. Therefore, SEND protocol is proposed to automatic protection of auto-configuration process. It is secure neighbor discovery and address resolution process. To defend against threats on NDP's integrity and identity, Cryptographically Generated Address (CGA) and asymmetric cryptography are used by SEND. Besides advantages of SEND, its disadvantages like the computation process of CGA algorithm and sequentially of CGA generation algorithm are considerable. In this paper, we parallel this process between network resources in order to improve it. In addition, we compare the CGA generation time in self-computing and distributed-computing process. We focus on the impact of the malicious nodes on the CGA generation time in the network. According to the result, although malicious nodes participate in the generation process, CGA generation time is less than when it is computed in a one-way. By Trust Management System, detecting and insulating malicious nodes is easier.

***Keywords*—**NDP, IPsec, SEND, CGA, Modifier, Malicious node, Self-Computing, Distributed-Computing.

I. Introduction

AFTER allocating last blocks of IPv4 address space and finishing IPv4 address space, transition to IPv6 has been happened. On 8 June 2011, World IPv6 Launch represents a major milestone in the global deployment of IPv6. For the first time**,** IPv6 is enabled permanently by ISPs and web companies on 6 June 2012. Migrating to IPv6 was indispensable but doing it in a way that avoids the security risks of ipv6 deployment is required. In IPv6 we do not have ARP like Ipv4 and it is replaced with ICMP based NDP protocol. ND protocol uses IPv6 ICMP messages to find and resolve neighbors IPv6 addresses [1]. Finding neighboring routers, determining the link-layer addresses for neighbors, finding reachability of neighbors and detecting changed link-layer addresses are some functions of NDP [2]. NDP has some critical functions but because it was designed to work in trustworthy links, it is vulnerable to some attacks such as Duplicate Address Detection DoS attack, Neighbor Unreachability Detection Failure, Parameter Spoofing attack, Bogus Address Configuration Prefix, Bogus On-Link Prefix, Malicious Last Hop Router attack, Kill the Default Router attack, Compromise of a Router, Spoofing of Redirect Messages, and Remote/Replay attacks [3]. To apply NDP in an untrusted network like public networks that a malicious user can forge NDP messages and generate attacks by impersonating legitimate nodes [4], a security mechanism is essential.

The security mechanism which was introduced to secure NDP was SEND [5]. SEND ensures message integrity, verifies routers' authority and prevents attacks. A SEND packet contains, CGA, RSA signature, Nonce, and Timestamp [4] options which are appended to the regular NDP message.

The CGA address is generated by a SEND-enabled node which generates or obtains a public and private key pair and the outgoing ND messages is signed. To accept the received message, a receiver node verifies it. If it is not successful, the message will be discarded.

Computing two independent one-way hash values (Hash1 & Hash2) are done in CGA address generation process. The output of Hash2 computation is the input of the Hash1 computation.

The proposed idea of this paper is paralleling CGA generation process on resources of a network in order to compute it. In this method, if nodes of network have single or multi core CPU, the time of computing CGA will decreases and the final modifier value is calculated faster in comparison to [6]. In this method, if a node is a malicious one, it is determined and its task of computing the final modifier will transfer to other nodes. Moreover, the impact of the malicious nodes on the CGA generation time in the network is considered.

The structure of this report is as follows. NDP is overviewed in Chapter II and Chapter III represents Secure NDP. Chapter IV describes Cryptographically Generated. Chapter V reviews previous works. The proposed method and its implementation are described in Chapter VI. Chapter VII shows the result of implementing the proposed method. Finally, Chapter VIII mentions the conclusion.

II. Neighbor Discovery Protocol

NDP is based on ICMPv6 protocol [7].

Mahnaz Moslehpour is with the Amirkabir University of Technology (Tehran Polytechnic), Iran (email: moslehpour@aut.ac.ir).

Siavash Khorsandi is with the Computer Engineering and Information Technology Department of Amirkabir University of Technology (Tehran Polytechnic), Iran (e-mail: khorsandi@aut.ac.ir).

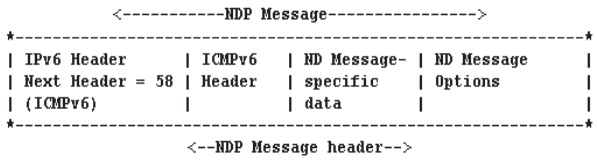

Fig. 1 NDP Message

NDP defines five ICMPv6 packet types, from 133 to 137, for the purpose of router solicitation, router, neighbor solicitation, neighbor advertisement, and network redirects. These messages are used to provide the following functionality:

- Router discovery: it is used to discover routers in an IPv6 network.
- Prefix discovery: hosts can discover address prefixes that are on-link for attached links.
- Parameter discovery: hosts can find link parameters like hop limits and MTU.
- Address autoconfiguration: stateless configuration of addresses of network interfaces.
- Address resolution: mapping between IP addresses and link-layer addresses.
- Next-hop determination: hosts can find next-hop routers for a destination.
- Neighbor unreachability detection (NUD): determine that a neighbor is no longer reachable on the link.
- Duplicate address detection (DAD): nodes can check whether an address is already in use.
- Packet redirection to provide a better next-hop route for certain destinations.

III. Secure Neighbor Discovery

As NDP is used by both hosts and routers, it is more vulnerable to various attacks unless secured. RFC 3971 specifies security mechanisms for NDP; unlike those in the original NDP specifications (Fig. 1), these mechanisms do not use IPsec. RFC 3971 specifies the Secure Neighbor Discovery protocol which is counters the threats to NDP and protects NDP messages. SEND is applied in environments where physical security on the link is not assured, and attacks on NDP are a concern. SEND secures the NDP messages and introduces a set of new options (Fig. 2). This specification introduces options, an authorization delegation discovery process, and an address ownership proof mechanism. SEND packet contains four options, CGAs, the RSA Signature option, Nonce, and Timestamp [4].

CGAs are IPv6 addresses where the Interface Identifier (IID) is the hash computation of public key and specific parameters CGA is applied to prevent NDP messages from threats [4].

To generate a CGA, public key of a node and a proper sec value is selected. After performing the standardized algorithm [8], the node gets an IID, associated with the key pair. This IID results from the first 64 bits of the SHA-1 hash function [9] applied over the data structure called CGA parameters. Finally, Subnet Prefix concatenates to the IID to build the CGA address. Fig. 3, shows this process. Then, the node signs this address.

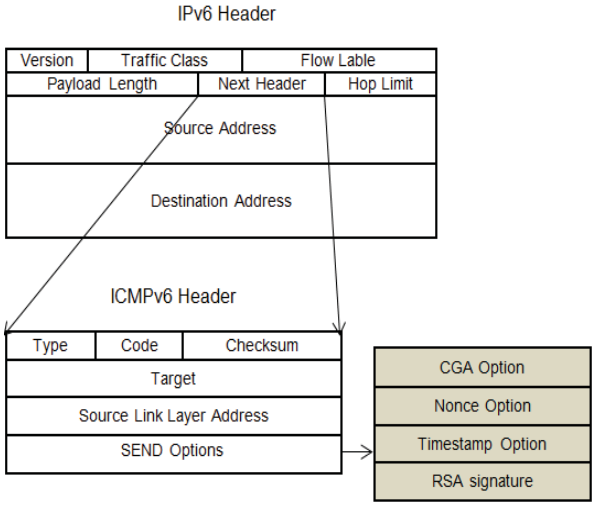

Fig. 2 NDP Message Protected by SEND

To verify a CGA address, an IPv6 node generates a CGA address by CGA parameters and public key of the new node. If it could regenerate the same CGA, it checks the validity of the signature to confirm the node using the CGA is the real owner of the public key related to this address.

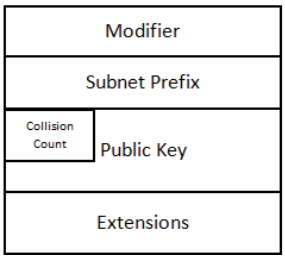

Fig. 3 CGA Parameters Data Structure

IV. Literature Review

Besides advantages of CGA algorithm to prevent stealing and spoofing the address, its disadvantages like high computational time of CGA generation encourage researchers to apply other solutions to optimize CGA algorithms. Reference [10] introduces a management mechanism for configuring CGA by hosts; in this technique, DHCPv6 which is automatically configuring hosts, is used in the CGA configuration.

Reference [11] is another technique which assumes the chance of address collision is small and while the normal DAD process is completed, a Tentative Address is used in communication messages. Reference [12] parallelizes Hash2 computation process because it is the heaviest part of CGA generation process. This technique increases the speed of CGA generation process. The last technique which is appropriate in mobile environment is [13]. It uses ECC keys instead of RSA keys because of the shorter key length.

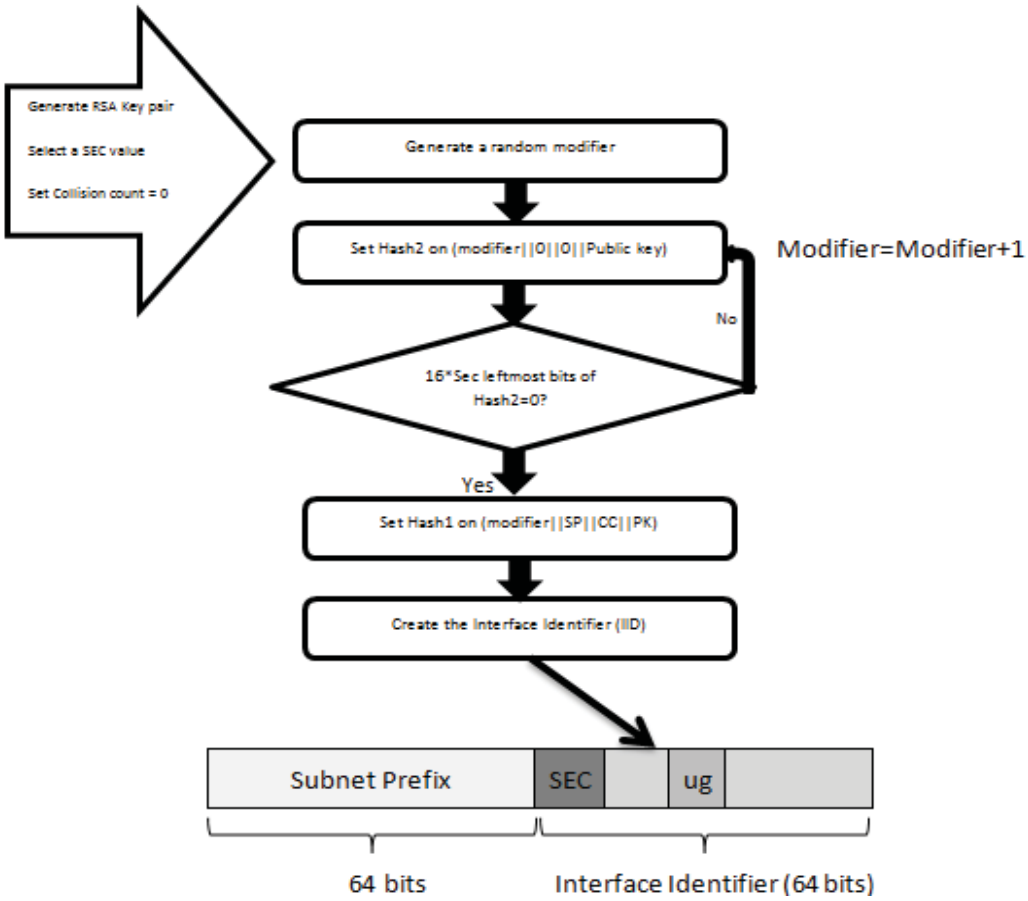

Fig. 4 CGA Generation Algorithm

V. Proposed Method

In previous researches, Hash2 computation was done by a new node which wants to communicate to a network. But, in this paper, we compute Hash2 value by utilizing the network resources and parallelizing the Hash2 computation process. In this case, we minimize the computational time of CGA algorithm. According to the idea, when the condition of 16 * Sec leftmost bits of Hash2 equals to zero is met, other nodes who participate in the computation process will be notified and they will stop their computation.

Because of using network resources, we concentrate on malicious nodes in an untrusted network because the probability of existing of malicious nodes is high; so, a system that can analyze behavior of nodes and assigns jobs for computing an appropriate final modifier in less time is efficient.

A. Trust Management

Trust Management is a system which detects, isolates malicious nodes and assigns jobs to nodes based on their behavior. It is used in an untrusted network where malicious nodes are available.

Detecting malicious nodes from un-malicious ones is done based on their responses and operations. This is done based on the verification steps of a returned value which are, 1) checking the conditions of 16*sec leftmost bit of a returned value must be equal to 0, and 2) checking a public key of the sender because a malicious node may alter the public key and then calculates a modifier. If each of these two verification steps fails, the calculated value is discarded [3].

When a node returns appropriate computed value, a value which passes two verification steps, it is a trusted node and it is labeled to White state node. If a node computes an appropriate value and sometime does not, it is labeled to Gray state node and when a node always computes inappropriate value, it is labeled to Black state. A node with this state is a malicious one and cannot trust it.

The state of Gray and Black nodes will be changed due to their operation. For instance, if a Gray state node computes an appropriate value, its state changes to white while if it computes an inappropriate value continually, it is changed to Black state [3].

The first function of Trust management is determining the state of participated nodes in the computation process; it is need because it helps to distribute task of computing the final modifier. The second function is, informing nodes about the state of other nodes due to prevent malicious nodes from disrupting un-malicious ones and removing the possible effect

of them on computation process. When a malicious node is detected, it stops computing and it will be isolated; in this case, its jobs are distributed to other nodes and while its state is not changed, job will not be assigned to it [3].

VI. Presentation, Analysis and Interpretation of Data

A. Implementation Scenarios

The tool which is applicable in all Windows Family [9] has been developed in Microsoft.Net as a service to provide security for Windows NDP and uses Winsock library to transfer data between network interface card (NIC) to upper layers and vice versa [7]. It does brute-force search to satisfy Hash2 condition of CGA algorithm in parallel. Based on the number of CPU cores of nodes, the numbers of parallel tasks which can be used for CGA computations are determined.

B. Extended Tool Testing

To evaluate the performance of CGA generation algorithm in parallel mode, several experiments are done on a base computer with 2.00 GHz CPU. The main operating system on this computer is Windows 7. We run the extended tool on guest windows 7 (32-bit) hosted by VMware Workstation 8.0 software. This process is generated 1000 times to have sufficient samples because it is a random process and no guaranties when to stop. All the measurements for different number of nodes which provides different number of cores are taken for CGA with Sec value "1" and with key size 1024-bit.

C. Test Evaluation

The extended tool is tested in different situation that the number of nodes is changed. Tables I-IV show the results of the CGA Generation Time and CGA Average Tries in two cases, self-computing and distributed-computing.

1.Self-Computing

In this case, the CGA generation address is a self-computed function. Table I demonstrates Average Generation Time and Average Tries of CGA generation process.

TABLE I
CGA Generation Time and Average Tries of CGA Generation Process in SELF-Computing State

Average Generation Time	Average Tries
336.614	66807.731

2.Distributed-Computing

In this case, CGA generation process is done by paralleling and distributing it between network resources. The proposed method is applied between different numbers of nodes.

According to Tables I-IV, by distributing the CGA generation process between network resources, the time of generation is decreased significantly. Whether in a case that dis-honest nodes are 60%, the time of generation when we apply the proposed method is less than self-computing. In addition, if the percentage of dis-honest nodes is zero, it means that all nodes trust each other and no nodes are malicious, the average generation time is in the best status and by distributing the process between 10 nodes it takes 213.016 milliseconds.

TABLE II
CGA Generation Time and Average Tries by Distributing the Process between 3 Nodes

Dis-Honest Nodes (Percentage)	Average Generation Time	Average Tries
0%	247.894	28334.089
10%	278.714	33362.338
20%	284.721	46117.479
30%	290.350	56469.137
40%	305.043	60181.193
50%	314.401	61205.155
60%	321.273	64680.578

TABLE III
CGA Generation Time and Average Tries by Distributing the Process between 6 Nodes

Dis-Honest Nodes (Percentage)	Average Generation Time	Average Tries
0%	231.806	12964.663
10%	260.714	18481.962
20%	280.771	25221.550
30%	288.810	37266.084
40%	295.817	45499.903
50%	300.078	52824.769
60%	313.370	57565.336

TABLE IV
CGA Generation Time and Average Tries by Distributing the Process between 10 Nodes

Dis-Honest Nodes (Percentage)	Average Generation Time	Average Tries
0%	213.016	11171.669
10%	247.381	13478.937
20%	253.910	16702.541
30%	257.082	22464.037
40%	264.944	32263.783
50%	269.675	35499.661
60%	273.430	48823.324

It is concluded that if the number of malicious nodes decreases and it approaches zero, the Generation Time and Tries would decrease.

Fig. 5 shows the comparison of CGA generation time in distributed-computing between three different number of node with different dis-honesty percentage. As it demonstrates, by increasing the number of participated nodes in CGA computation process, the generation time decreases. by increasing the number of malicious nodes, the generation time is increased and we can conclude that if network resources in generation process increase, the generation time will decrease.

Fig. 6 shows the comparison of self-computed with distribute-computed in a case that there are no malicious nodes; by applying the proposed method the generation time computes 123.598 milliseconds faster.

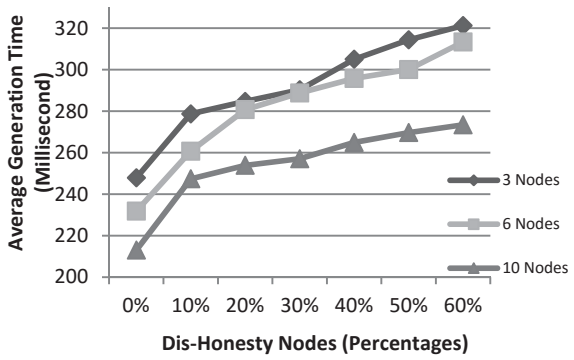

Fig. 5 The Comparison of Distributed-Computing in Three Different Number of Node

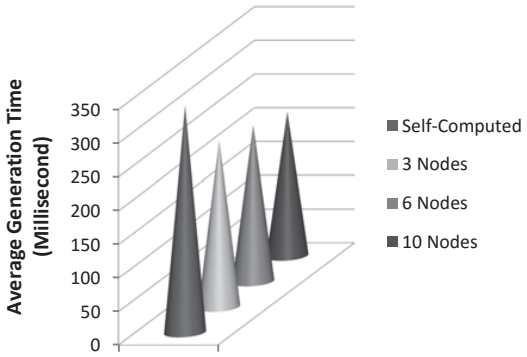

Fig. 6 The Comparison of Self-Computing and Distributed-Computing in 0% dis-honest nodes

VII. Conclusion

The goal of this research is to decrease CGA generation time by distributing this process between network resources; so, CGA computation cost is minimized and is made cost-effective. In addition, we concentrate on the effect of malicious nodes on CGA generation time. Based on results which are derived from different experiences, we can conclude that although malicious behavior has negative impacts on CGA generation process in distributed-computing method, CGA computational time is less than self-computing CGA generation process.

References

[1] Neighbor Discovery Protocol, http://howdoesinternetwork.com/2012/ndp-ipv6-neighbor-discovery-protocol.

[2] T. Narten, E. Nordmark, W. Simpson, Neighbor Discovery for IP version 6 (IPv6), RFC2461, Internet Engineering Task Force, Dec. 1998.

[3] M. Moslehpour, S. Khorsandi, Improving Cryptographically Generated Address Algorithm in IPv6 Secure Neighbor Discovery Protocol through Trust Management," In Press, 18th International Conference on Information and Communications Security (ICICS), Jun, 2016.

[4] A. AlSa'deh, H. Rafiee, and C. Meinel, Secure Neighbor Discovery: A Cryptographic Solution for Securing IPv6 Local Link Operation, Theory and Practice of Cryptography Solutions for Secure Information Systems Book, May, 2013.

[5] J. Arkko, J. Kempf, B. Zill, and P. Nikander, Secure Neighbor Discovery (SEND), RFC 3971, Internet Engineering Task Force, Mar. 2005.

[6] H. Rafiee, A. AlSa'deh, and C. Meinel, Multicore-Based Auto-Scaling Secure Neighbor Discovery for Windows Operating Systems, 26th IEEE International Conference on Information Networking (ICOIN), Feb. 2012.

[7] A. Conta, S. Deering, M. Gupta, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, RFC 4443, Internet Engineering Task Force, March 2006.

[8] T. Aura (2005), Cryptographically Generated Addresses (CGA) (Online). Available: http://tools.ietf.org/pdf/rfc3972.pdf

[9] OS Platform Statistics, http://www.w3schools.com/browsers/browsers_os.asp, 2011.

[10] Jiang, S., & Xia, S. (2012). Configuring cryptographically generated addresses (CGA) using DHCPv6.Retrieved from http://tools.ietf.org/html/draft-ietf-dhc-cga-config-dhcpv6-02.

[11] A. Moore, N. (2006). Optimistic duplicate address detection (DAD) for IPv6.Retrieved from http://tools.ietf.org/html/rfc4429.

[12] H. Rafiee, A. AlSa'deh, and C. Meinel, Multicore-Based Auto-Scaling Secure Neighbor Discovery for Windows Operating Systems, 26th IEEE International Conference on Information Networking (ICOIN), Feb. 2012.

[13] T. Cheneau, T. Boudguiga, A., & Laurent, M. (2010). Significantly Improved Performance of the Cryptographically Generated Addresses thanks to ECC and GPGPU. Computers & Security, 29(4), 419-431. Doi;10.1016/j.cose.2009.12.008.