Published as: Hosnieh Rafiee, Ahmad AlSa'deh and Christoph Meinel, "Multicore-Based Auto-Scaling SEcure Neighbor Discovery for Windows Operating Systems", 26th IEEE International Conference on Information Networking (ICOIN), Bali, Indonesia, February 1 - 3, 2012

Multicore-Based Auto-Scaling SEcure Neighbor Discovery for Windows Operating Systems

Hosnieh Rafiee, Ahmad AlSa'deh, and Christoph Meinel
Hasso-Plattner-Institut, University of Potsdam
P.O. Box 900460, 14440 Potsdam, Germany
{Hosnieh.Rafiee, Ahmad.Alsadeh, Christoph.Meinel}@hpi.uni-potsdam.de

***Abstract*—SEcure Neighbor Discovery (SEND) is proposed to counter IPv6 Neighbor Discovery Protocol (NDP) security threats. However, SEND is compute-intensive. Fulfilling Hash2 condition in Cryptographically Generated Addresses (CGA) is the main heavy part of SEND. Unfortunately, CGA computation cannot see significant speed improvement when it runs on multicore machine because CGA generation algorithm is sequential. In this paper, we propose a multicore-based high performance SEND implementation for Windows families to speed up SEND computations. The proposed approach automatically detects the number of processors available on a machine and creates equivalent number of working threads to compute Hash2 condition. The parallelization mechanism is implemented to assign CGA computation to all the cores. When one thread satisfies CGA Hash2 condition, the others stop. With the parallel approach, the speedup time has been increased extremely by increasing the number of cores in the computing device. Besides the parallelization, we extend SEND implementation to generate the key pair for CGA algorithm on-the-fly to enhance the security and to protect the privacy.**

Keywords-SEND implementation; Cryptographically Generated Addresses (CGA); Neighbor Discovery Protocol (NDP); IPv6 security and protection; parallel computing

I. INTRODUCTION

SEcure Neighbor Discovery (SEND) [1] is as an extension to Neighbor Discovery Protocol (NDP) [2, 3]. SEND uses RSA key pairs, Cryptographically Generated Addresses (CGA) [4], digital signature and X.509 certification to offer significant protection to NDP. SEND offers the address ownership proof, message integrity and router authorization mechanism.

Unfortunately, SEND is computationally heavy especially for high security level requirements. The sequential brute-force search loop to fulfill Hash2 condition [4] in CGA generation algorithm is the main compute-intensive part of SEND. It may take several hours or even days to find CGA parameters with security level "2". This long delay is unacceptable in several applications.

Nowadays, multicore processors greatly increase the computational capacity of computers and become standard. Nearly every computer has a processor with at least two cores. Some desktops have up to 8 CPUs. These CPU numbers are likely to increase in the future. Multicore processor offers the opportunities for remarkable speed improvement in CGA computation. However, a dual-core processor does not double CGA computation performance automatically, because CGA generation algorithm is sequential process and not fully utilized the available CPU cores. The conventional implementation of CGA does not take full advantage when it runs on a multicore machine. Therefore, exploiting the computational resources of the existing cores requires CGA generation algorithm improvement to run in parallel. So, it is critical that CGA implementation scale with the number of cores.

In this paper, we will introduce an approach for multicore device to achieve better performance for SEND computations. We extend Windows SEcure Neighbor Discovery (WinSEND) [5] implementation to do the brute-force search to find a valid modifier in parallel rather than do it sequentially. Based on the number of cores on the computing device, the extended version of WinSEND determines the number of parallel threads to compute CGA address parameters. The parallel WinSEND version reduces the CGA generation time drastically. The experiment results are validated and evaluated by comparing the performance of the proposed approach with the conventional approach. Besides, we extend WinSEND to generate the key pair for CGA algorithm on-the-fly. Generating the keys in spot increase the randomness of CGA addresses and enhances its security against brute force attack and protects the users' privacy.

The paper is structured in the following manner. An overview of NDP and possible attacks against it is presented in Section II. Section III shows how SEND can protect NDP and provides more details about CGA generation algorithm. Section IV explains the details of parallelization of SEND implementation and shows new extensions which is added to WinSEND implementation. The experimental results are demonstrated in Section V. The last section concludes the work.

II. NEIGHBOR DISCOVERY PROTOCOL (NDP)

Neighbor Discovery (ND) for IPv6 [2] and IPv6 Stateless Address Auto-Configuration (SLAAC) [3], together, are referred to as IPv6 Neighbor Discovery Protocol (NDP). NDP is one of the main protocols in IPv6 suite. It greatly improves the efficiency and the network management. It is heavily used for several critical functionalities, such as discovering other existing nodes on the same link, determining others' link layer addresses, detecting duplicate addresses, finding routers and maintaining reachability information about paths to active

neighbors. Moreover, NDP plays a crucial role in mobile IPv6 (MIPv6) networks. In MIPv6, the SLAAC and ND features eliminate the need for Foreign Agents (FAs), and a Mobile Node (MN) can join to a new foreign network by receiving the Router Advertisements (RAs) [6].

NDP has only basic protection mechanisms to ensure that the messages come from nodes that directly connect to a local link. ND accepts messages which come from nodes with either unspecified or link-local IPv6 addresses and with hop limit 255. However, this protection shield is not enough to protect IPv6 local networks. If NDP messages left without authentication mechanism, it will be vulnerable to a set of attacks. IPv6 Neighbor Discovery (ND) Trust Models and threats, RFC3756 [7], describes these attacks. The attacker can carry out several attacks based on address resolution, redirect, Duplicate Address Detection (DAD), Router Advertisement (RA), and address configuration. For instance, it is easy to configure a rogue router on the link, and it is difficult for the host to validate the authorized router.

In addition the security weakness, StateLess Address Auto-Configuration (SLAAC) leads to privacy implications. Since generating the interface identifier from the MAC address of the node (which remains constant over time) makes it possible to track a node over the Internet. Also, SLAAC make it easy to correlate the traffic patterns and the activities to the certain user [8].

III. SECURE NEIGHBOR DISCOVERY (SEND)

A. SEND Options and Messages

SEcure Neighbor Discovery (SEND) [1] is a set of enhancements to NDP. SEND offers three additional features to NDP: Address ownership proof, message protection and router authorization mechanism. To achieve these enhancements, SEND comes with four new options: `CGA`, `RSA Signature`, `Timestamp`, and `Nonce` options.

1. CGA Option: this field carries the associated parameters to enable the receiver to validate the proper binding between the owner public key and the Cryptographically Generated Address (CGA).
2. RSA Signature Option: this option is used to authenticate the identity of the sender. The message which is sent from CGA address is signed with the address owner private key and the public key is used to verify the signature. This signature prevents an attacker from spoofing NDP messages.
3. Nonce Option: this option is used to protect messages from replay attacks, and to ensure that an advertisement is a fresh response to a solicitation which is sent earlier by the node.
4. Timestamp Option: this option is used to protect the unsolicited advertisements (periodic RA and Redirect messages) from replay attacks.

SEND uses an Authorization Delegation Discovery (ADD) process to validate and authorize IPv6 routers to act as default gateways, and specifies the IPv6 prefixes that a router is authorized to announce on the link [1]. ADD relies on an electronic certificate issued by a trusted third party. Before any node can accept a router as its default router, the node must be configured with a trust anchor(s) that can certify the router via certificate paths. Thus, the node requests the "router" to provide its X.509 certificate path to a Trust Anchor (TA) which is preconfigured on the node. The "router" should not be trusted if it fails to provide the path to TA. Two new ICMPv6 discovery messages are offered for identifying the router authorization process:

1. Certificate Path Solicitation (CPS): is sent by hosts during the ADD process to request a certification path between a router and one of the host's trust anchors.
2. Certificate Path Advertisement (CPA): is sent in reply to the CPS message and contains the router certificate.

B. CGA Generation Algorithm

Cryptographically Generated Address (CGA) is an essential part of SEND which is proposed to prevent address stealing. It offers the authentication to IPv6 addresses without the need of the third party or additional security infrastructure. CGAs are IPv6 addresses, where the interface identifiers (IIDs) are generated by one-way hashing of the node's public key and other auxiliary parameters. Thus, the IPv6 address of a node is bound to its public key. This binding can be verified by re-computing the hash value and being compared with the interface identifier of the sender IPv6 address.

In CGA generation algorithm, the address owner computes two independent one-way hash values (Hash1 and Hash2) by using the public key and other auxiliary parameters. Hash2 value sets an input parameter for Hash1. The combination of the two hash values increases the computational complexity of generating new address and the cost of brute-force attacks.

A schematic of CGA generation algorithm is shown in Figure 1 (refer to RFC 3972 for more details). CGA generation begins with determining the address owner's `Public Key` and selecting the proper Security level (Sec) value. Then continue the Hash2 computation loop until finding the `Final Modifier`. Hash2 value is a hash of combination of the Modifier and the `Public Key` is concatenated with zero-value of `Subnet Prefix` and `Collision Count`. The address generator tries different values of the Modifier until 16×Sec-leftmost-bits of Hash2 computes to zero. Once a match is found, the loop for Hash2 computation terminates. Afterward, the Final Modifier value is saved and used as an input for Hash1 computation. Hash1 value is a hash of combination of the whole CGA parameter data structure. Then, the interface identifier (IID) is derived from Hash1. The Sec value is encoded into the three leftmost bits of the interface identifier. The 7th and 8th bits from the left of IID are reserved for special purpose (Refer to RFC 3513 for more details). Finally, the Duplicated Address Detection (DAD) is done to ensure that there is no address collision within the same subnet.

The main disadvantage of using CGA is the computational cost. CGA computations may take long time especially for high "Sec" value. In fact, satisfying Hash2 condition is the most computational expensive part of CGA generation algorithm. Multiple hashes are computed over the CGA parameter data

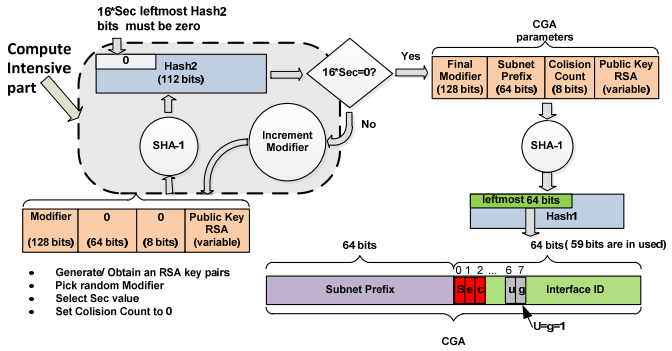

Figure 1. CGA generation algorithm.

structure which is slightly modified each time the modifier value at once. One approach to speedup CGA computations is to parallelize CGA algorithm.

C. *Parallelization of CGA Generation Algorithm*

CGA generation is a sequential algorithm. Therefore, the main challenge of CGA parallelization is to break down CGA algorithm in parallel and assemble it to find the final results. To parallelize CGA algorithm, two things should be determined: which part of CGA algorithm should be parallelized and to how many parallel process it should be broken. Hash2 condition needs to be broken to number of threads because it is the heaviest part of CGA. To utilize all cores on a device, the number of threads is determined based on the number of CPU cores. In this way, the CGA generation algorithm can scale its performance based on the available CPU cores. It is important to understand that using two processors do not double the speed. Parallel processing causes some level of overhead, the resources used to manage the multiple processors.

IV. PARALLELIZED SEND IMPLEMENTATION

We implement SEND parallel mode by extending Windows SEcure Neighbor Discovery (WinSEND) [5] implementation. More details about WinSEND implementation and the extensions to achieve the parallelization and other features are shown in the following subsections.

A. *WinSEND Implementation Overview*

WinSEND has been developed in Microsoft .NET as a service to provide security for Windows NDP. WinSEND is proposed as a response to the lack of SEND implementation in Windows XP/Vista/7 [11]. While Windows family is the most popular operating system and accounts for more than 80% of usage compared to other operating systems [12]. It uses `WinPcap` library [9] rather than Winsock to transfer data between network interface card (NIC) to upper layers and vice versa. WinSEND has direct access to raw sockets which offers the possibility for an application to receive/send network traffic before processing it by the normal TCP/IP stack.

WinSEND is subdivided into 3 main components: User Interface, WinSEND Service and Main Classes. Figure 2 shows WinSEND components and the relationship between them. WinSEND Main Classes is a shared component that is called by both WinSEND Service and User Interface. Users

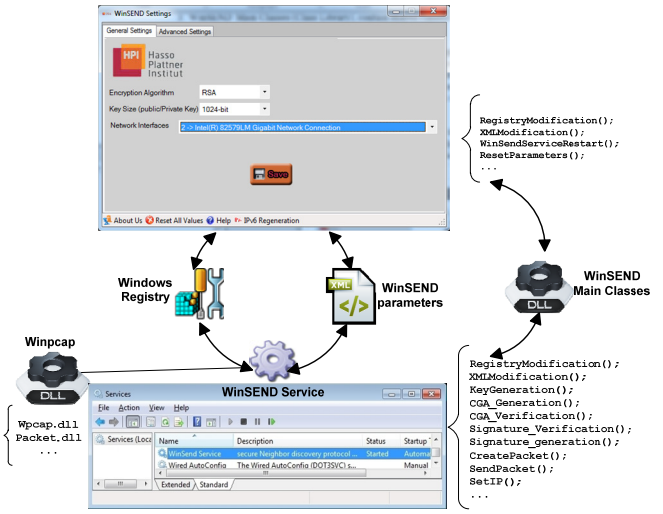

Figure 2. WinSEND main components and its relationships

can select their desired parameters such as security level (Sec value), cryptosystem, key size and network interface card which will be secured by WinSEND. These configuration parameters are stored in an XML file format for further usage. This XML file contains SEND options such as the "subnet prefix", "Sec" value and key pair (public/private keys). The reason for storing CGA parameters is to skip CGA generation process while a node is connected to the same network or the address is still valid. The address becomes invalid once the node joins a new subnet or the valid time is elapsed. For generating new CGA addresses, WinSEND reruns all CGA generation processes automatically and update the XML file. WinSEND uses `netsh` [10] command to disable the NDP messages (auto-configuration, router discovery, and etc) from normal TCP/IP stack. In this way, the exchange of NDP message is forced to be handled by WinSEND in secure manner.

B. *WinSEND Parallel Computing Mode Implementation*

WinSEND is extended to do the brute-force search to satisfy Hash2 condition of CGA algorithm in parallel. In parallel computational mode, the extended WinSEND can use almost the whole CPU capacity to finish CGA computations. When the extended WinSEND starts, it reads the number of CPU cores and base on the number of existing cores, WinSEND determines the number of parallel tasks which can be used for CGA computations.

Figure 3 shows how CGA generation can be parallelized to use n-core computing device. In this approach, each task is run in one core to avoid CPU switching among the parallel tasks for CGA generation. The Tasks Generator (TG) passes a particular continuous sets of modifier for each task, i.e. each thread dedicated to do brute-force search for certain range. If the thread1 process modifier range from `0` to `n`, then thread2 would process for a range of `n+1` to `2*n` and so on. TG also passes termination token to CGA Generation (CGAGen) function to control and terminate all tasks when one of the tasks succeed to find a valid modifier which fulfills Hash2 conditions. If none of the tasks successes to find the valid

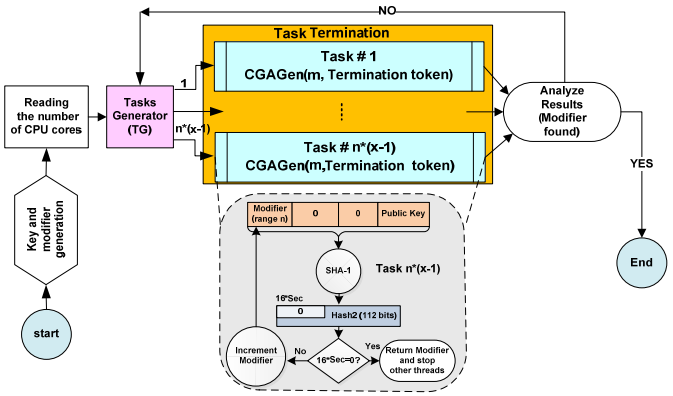

Figure 3. Flowchart of the Proposed Parallel CGA generation processing

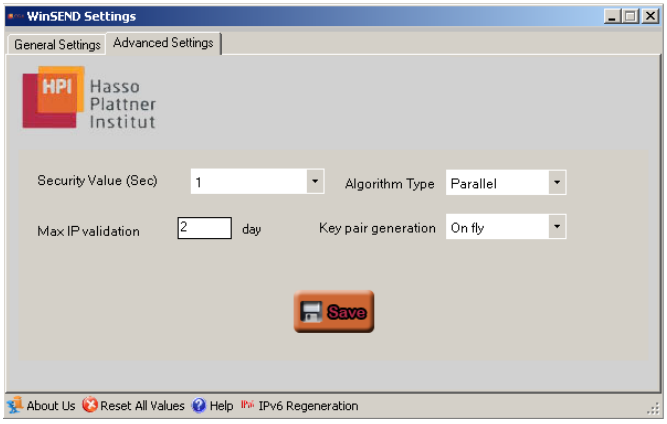

Figure 4. WinSEND Interface advanced setting

modifier within the assigned range, new tasks with new ranges of modifiers are generated.

WinSEND uses some customized classes in Task Parallel Library (TPL) [13] to simplify the parallelization management. TPL simplifies this process and provides a number of classes and methods which allows running a number of jobs in a parallel fashion. TPL has Task Factory class [14] that provides support for creating and scheduling Task objects without sacrificing of their power and flexibility. The following mathematic formula shows total time requires meeting Hash2 condition in CGA sequential and parallelizing algorithm.

- Sequential CGA algorithm

$$t_{seq}(m) = t_o + \sum_{i=0}^{r} t_{h2l}(m+i)$$

t_{seq} = Total time needs for forming the final CGA address in sequential mode

t_o = The time needs for other steps in CGA generation algorithm, such as RSA key generation, Hash1 calculation, and Duplicate Address Detection (DAD) check.

t_{h2l} = The time needs for computing Hash2 loop.

m = Modifier

r = The total number of iteration to fulfill Hash2 condition

- Parallelize CGA algorithm

$$t_{par}(m) = t_o + \sum_{i=0}^{r} min_i \begin{pmatrix} \sum_{j=0}^{n} t_{h2l}(m+j) \\ \sum_{j=n}^{n*2} t_{h2l}(m+j) \\ \vdots \\ \sum_{j=n*(x-1)}^{n*x} t_{h2l}(m+j) \end{pmatrix}$$

t_{par} = Total time needs for processing CGA in parallel mode

n = Constant number

x = Number of CPU cores

Nevertheless, the user can change the computational mode to sequential mode via the advance setting in WinSEND User Interface as shown in Figure 4. In sequential computation mode, one instruction is executed per unit of time. The sequential WinSEND cannot use the total CPU capacity of all cores on a computing device. The default setting of WinSEND uses the parallel computational mode.

C. *Key Generation Method*

When WinSEND service runs on a node, it calls "`KeyGeneration`" function that generates key pairs based on a user-defined key size automatically (by default 1024-bit). Generation the keys in this way has the following advantages:

- The user requires minimal amount of configuration, no need for her/him to know the technical details behind the cryptography.
- No need to use external program to generate the key pairs. In practice, it is not easy for the user to generate key pairs manually each time the application wants to generate new address for the computing device.
- Keys are not stored in a particular path before starting the application. Therefore, the keys are not vulnerable to stealing.
- Generating the key pair on-the-fly each time the WinSEND starts CGA process, enhances the CGA security and protects the user privacy. Generating the `public key` in this way increases the randomness of CGA generated addresses and consequently enhances its security against the brute-force attack. Moreover, each time the node moves to a new location, it gets new CGA address and uses new public key. Therefore, it is not easy to track the users based on their IPs or even to correlate the traffic to their public keys.
- The average time to generate key pair with RSA key 1024-bit over 1000 sample cost 27.8 Milliseconds, while the average time for CGA generation for Sec

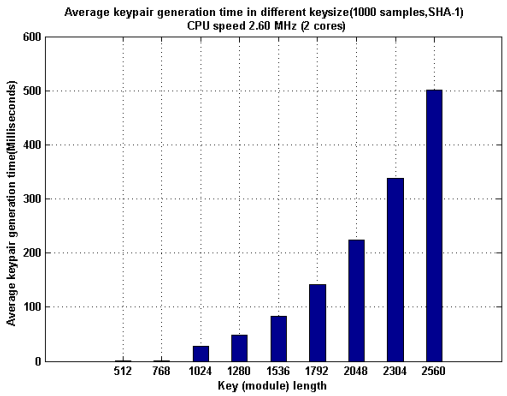

Figure 5. Average key pair generation time for different key (module) length over 1000 samples

value "1" and key 1024-bit cost 439.6 Milliseconds. So, the key generation takes around 6.3% of the total CGA generation time. The measurements are done on a computer with 2.6 GHz. Figure 5 shows this average time generation mapped to different key sizes.

- Reading 1024-bit key sizes from a PEM file and convert it to a readable format for WinSEND takes almost the same time if the key is generated in spot.

V. EXTENDED WINSEND TESTING

To evaluate the performance ofCGA generation algorithm in parallel mode, several experiments are carried out for different system specifications. The experiments are done on a computer with 2.60 GHz CPU (8-cores) and 8GB RAM. Debian is the main operating system on this computer. We run the extended WinSEND on guest Windows 7 (64-bit) hosted by Virtualbox4.1.0 software. The settings of VirtualBox offer the flexibility to control the number of virtual CPU cores that the guest operating systems can use. Because the CGA generation is a random process and no guarantees when to stop, the CGA address is generated 1000 times to have sufficient samples. All the measurements for different number of cores are taken for CGA with Sec value "1" and with key size 1024-bit.

Figure 6 shows the CPU usage for WinSEND parallel and sequential modes. In sequential mode, the CPU usage around 30% while in parallel mode it is around 80% of the total CPU capacity. In recent Windows versions (vista, 2008 and 7), multi-core CPUs is supported natively and the OS attempts to distribute a job (sequential or Parallel) over CPU cores and decrease execution time. However, this distribution is not as efficient as a developer uses the parallelization techniques in the application. In addition, the thread limits affects the execution time too. The thread limits in 32-bit and 64-bit Windows 7 varies [15] i.e. In 32-bit Windows 7 with 2GB RAM, about 2025 concurrent threads can be created while in 64-bit windows is 54965 threads. Therefore, WinSEND sequential mode performs better for 64-bit OS than 32-bit.

The experimental results show considerable speedup with parallel approach, compared to the conventional sequential

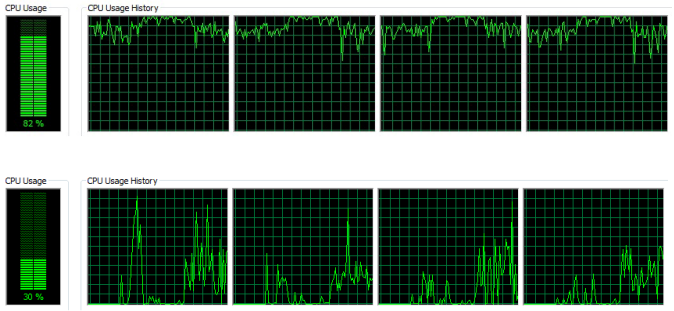

Figure 6. CPU usage snapshot for 4-cores CPU. Top-WinSEND parallel mode; Bottom- WinSEND sequential mode

approach. Table 1 shows CGA average generation times for parallel and sequential WinSEND. The measurements are taken for CGA with Sec value "1" and for RSA public key 1024-bit. With 2 cores, the parallel approach speedup CGA computation by 27.1%, for 4 cores by 30.5% and for 8 cores by 38.7%. It is clear that CGA benefit from increasing the number of cores available on a computer in both cases with sequential and parallel modes. However, the parallel mode is much better than sequential because it tries to invest most of the available CPU capacity to do the computations.

TABLE I. CGA AVERAGE GENERATION TIME COMPARISON OF SEQUENTIAL AND PARALLEL WINSEND

Number CPU cores	CGA average generation time (Milliseconds) 1024-bit RSA key, Sec=1		Percentage of Speedup
	Parallel Mode	Sequential Mode	
2	376.34	516.26	27.1%
4	304.13	437.82	30.5%
8	261.43	426.36	38.7%

VI. CONCLUSION

Secure Neighbor Discovery (SEND) protocol is proposed to counter most of the threats against Neighbor Discovery Protocol (NDP). Unfortunately, SEND is compute-intensive approach. Since SEND is sequential algorithm, it cannot see the advantage automatically when it runs on multicore machine. It cannot use all CPU capacity. Therefore, to benefit from multicore technology, there is a need to redesign the implementation of programs and software to parallelized model.

In this paper, we presented a multicore-based parallel computational approach for CGA which is significantly improve CGA generation time. This approach can automatically scale based on the number of available cores on the computing device. The experimental results show considerable speedup with CGA generation time in parallel mode compare to the sequential approach. The amount of speedup achieved depends on how many cores are available. This improvement enables the device to have "more secure" address within shorter time.

Windows SEcure Neighbor Discovery (WinSEND) implementation is extended to work in parallel mode. WinSEND is developed in Microsoft .NET. It can be installed in windows as a service. It has an easy user interface to enable the user to set the desired security parameters. WinSEND uses WinPcap API to have a direct access to raw sockets and bypass normal TCP/IP stack.

REFERENCES

[1] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[2] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[3] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[4] T. Aura, "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005. Updated by RFCs 4581, 4982

[5] H. Rafiee, A. Alsa'deh, aand Ch. Meinel, "WinSEND: Windows SEcure Neighbor Discovery", 4th International Conference on Security of Information and Networks (SIN 2011), 14-19 November 2011, Sydney, Australia. 2011.

[6] R. Ed. Koodli, "Mobile IPv6 Fast Handovers", RFC 5568, July 2009.

[7] P. Nikander, J. Kempf, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004

[8] T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[9] Winpcap documentation, http://www.winpcap.org

[10] Microsoft TerchNet, Netsh Technical Reference, http://technet.microsoft.com/en-us/library/cc725935, 2009.

[11] Microsoft TechNet, IPv6 Security Considerations and Recommendations, http://technet.microsoft.com/en-us/library/bb72695, 2011.

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

[13] Microsoft TechNet, Task Parallelism (Task Parallel Library), http://msdn.microsoft.com/en-us/library/dd537609.aspx, 2011.

[14] Microsoft TechNet, Task Factory, http://msdn.microsoft.com/en-us/library/system.threading.tasks.task.factory.aspx

[15] M. Russinovich, D. Solomon, and A. Ionescu, Windows Internals Book, 2009.