Published as: Ahmad AlSa'deh, Hosnieh Rafiee and Christoph Meinel, "Stopping Time Condition for Practical IPv6 Cryptographically Generated Addresses", 26th IEEE International Conference on Information Networking (ICOIN), Bali, Indonesia, February 1 - 3, 2012.

Stopping Time Condition for Practical IPv6 Cryptographically Generated Addresses

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

Abstract— **Cryptographically Generated Addresses (CGA) are employed as an authentication mechanism in IPv6 network to realize the proof of address ownership without relying on any trust authority. The security parameter (Sec) indicates the security level of the CGA address. For Sec value greater than zero, there is no guarantee to stop the brute-force search after certain time. The address generator tries different values of Modifier until (16×Sec)-leftmost-bit of the second hash (Hash2) computes to zero. This paper proposes some modifications to the standard CGA "RFC 3972" in order to limit the time that CGA generation may takes. The modified CGA generation algorithm takes the upper bound of CGA running time as an input and the Sec value is determined as an output of the brute-force computations. The modified CGA keeps track of the best founded Hash2 value during the running time. The paper also proposes to reduce the granularity of the security level from "16" to "8", to increase the chance to have better Sec value within the time limit. We called the modified CGA as Time-Based CGA (TB-CGA). The implementation and evaluation of TB-CGA are done in this paper.**

Keywords-IPv6 security; SEcure Neighbor Discovery; CGA performance

I. INTRODUCTION

Cryptographically Generated Addresses (CGA) [1] are designed to offer the authentication to IPv6 addresses and prevent malicious nodes from claiming the ownership of the others' addresses. CGA is an IPv6 address where the interface identifier part is generated from a cryptographic hash of the address owner's public key and other parameters. Thus, IPv6 address of the node is bound to its public key. In this manner, CGA is self-certifying since it does not rely on Public-Key Infrastructure (PKI) or other authority. Therefore, any IPv6 node can generate its CGA address locally.

CGA authenticates the identity of the sender based on public key cryptography. The recipient is able to determine that the message comes from a real sender. The message which is sent from CGA address is signed with the address owner private key and the public key is attached to the signed message. Since the message contains everything the recipient needs to authenticate, the receiver does not need to have further communications with the sender for completing the authentication process. The receiver authenticates the message by verifying that the hash of the public key matches the sender's address and the signature is valid.

For using CGA, the sender node needs to select the CGA Security Parameter (Sec). Sec value indicates the security level of the CGA against the brute-force attacks. Sec is an unsigned 3-bit integer having a value between "0" and "7". It increases the computational cost for both the attacker and the address generator. The address generator needs, on average, $2^{16\times Sec}$ brute-force search to satisfy Hash2 condition [2], i.e., (16×Sec)-leftmost-bit of Hash2 equal to zero. Large Sec value may leads to significant and undesirable address generation delay. For Sec value "2", the CGA address computation takes several hours on a computer with 2.67 GHz CPU speed. Currently, it is impractical to use CGA with large Sec value especially in recourse-constrained networks, such as in mobile telephones, wireless sensors, and ad-hoc networks where nodes have limited resources (battery, memory, processor, and bandwidth). The CGA computation will take too long time and consumes the computing device energy. Therefore, the high computation cost of CGA may prevent its usage and leave IPv6 networks vulnerable to several attacks which are related to address stealing.

Normally, the address owner sets Sec value. But it is hard for the user to select the suitable Sec value. Small Sec value leaves a small margin of safety and large Sec value may causes unacceptable address generation delay. Even though in case the user knows the details of CGA algorithm, it is hard to predict the CGA generation time because the computation of Hash2 is completely random, and it is not easy to predict the required CGA generation time for Sec value greater than "0". Moreover, CGA generation time depends on computing device CPU speed. Consequently, it is better to select Sec value in more practical way base on tangible factor that the user can determine it, such as time.

We propose a modified CGA with termination time to force CGA generation to stop after certain time specified by the user or the address generator. Our purpose is to gain an optimal advantage from CGA security without waiting long time for CGA generation. The modified CGA generation algorithm takes the termination time as an input and then determines Sec value as an output of CGA computation. In this paper, we propose the following modifications to the standard CGA [1]:

- Select time parameter as an input instead of Sec value. The time parameter is set to ensure that CGA will stop after certain time. The Sec value is determined by

rounding down the number of zeros in the best founded Hash2 value to the nearest multiple factor "8".

- Replace the standard granularity factor "16" with "8" in Hash2 condition to get an optimal security level within the stopping time and reduce the number of wasted iterations to find better Sec value. It is obvious that the chance to have hash function output with "8" successive zero is higher than it with "16".

The paper is structured as follows. Section II reviews CGA generation algorithm. Section III discuses the CGA modifications decision to reach the final time-based CGA (TB-CGA) generation algorithm. Section IV describes the testing environment and the TB-CGA implementation. Section V shows some measurements for standard CGA and the modified version (TB-CGA). Section VI concludes the paper.

II. CRYPTOGRAPHICALLY GENERATED ADDRESSES (CGA)

CGA firstly proposed as a mechanism for authenticating location updates in Mobile IPv6 [3]. Later, CGAs were standardized in the context of the SEcure Neighbor Discovery (SEND) [4] to protect Neighbor Discovery (ND) for IPv6 [5] and IPv6 Stateless Address Autoconfiguration [6] against known attacks [7]. CGA is also proposed to prevent Denial-of-Service (DoS) attack and to authenticate the Binding Update messages in Mobile IPv6 [8, 9].

A. CGA Generation Algorithm

In CGA, the interface identifier portion of IPv6 address is created from cryptographic hash of the address owner's public key and other auxiliary parameters. Since the 64-bit are not enough to provide sufficient security against brute-force attacks in the foreseeable future, the standard CGA uses the Hash Extension [1] to increase the security strength above 64-bit.

The purpose of the Hash Extension [2] is to increase artificially the cost of creating a hash without increasing its length. The address owner computes two independent hash values (Hash1 and Hash2) by using the public key and other parameters. The Hash Extension (Hash2, or portion of it) 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. CGA generation algorithm should fulfill two conditions [1]:

1. The leftmost 64-bit of Hash1 equals the interface identifier. The Sec, "u" and "g" bits are ignored in the comparison.
2. The 16×Sec leftmost bits of Hash2 are equal to zero.

The security parameter (Sec) indicates the security level of the generated address against the brute-force attacks. Increasing Sec value by "1" adds 16-bit to the length of hash that the attacker must break. Sec is an unsigned 3-bit integer having a value between "0" and "7".

The use of CGA requires the sender to send CGA parameters to the receiver. CGA parameters are concatenated to form a CGA parameter data structure which contains the following parameters:

1. Modifier (128-bit): is initialized to random value.

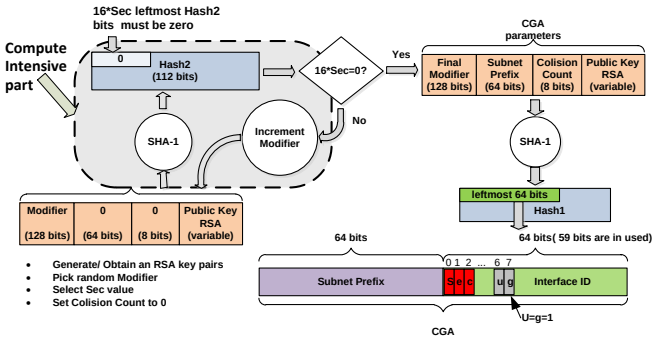

Figure 1. CGA generation algorithm.

2. Subnet Prefix (64-bit): it is set to routing prefix value advertised by the router at the local subnet.
3. Collision Count (8-bits): is a collision counter used for Duplicate Address Detection (DAD) to ensure the uniqueness of the generated address.
4. Public Key (variable length): is set to the DER-encoded public key of the address owner.
5. Extension field has variable length for future needs.

A schematic of CGA generation algorithm is shown in Figure 1. CGA generation begins with determining the address owner's public key and selecting the proper 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 hash value is truncated to the appropriate length (64-bit). 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. Finally, the Duplicated Address Detection (DAD) is done to ensure that there is no address collision within the same subnet.

B. CGA Generation Computational Cost

CGA algorithm increases the computational cost for both the attacker and the address generator (owner). The address generator needs $O(2^{16\times Sec})$ brute-force search to satisfy Hash2 condition and finding the Final Modifier. The attacker needs to do a brute-force attack against an (16×Sec+59)-bit hash value which costs $O(2^{16\times Sec + 59})$.

Fulfilling the condition of Hash2 is the computationally expensive part of CGA generation. The address owner may have not powerful machine to compute CGA within certain time. Selecting too high Sec value may cause unacceptable delay in address generation. For Sec value greater than zero, there is no guarantee to stop after a certain number of

iterations. Therefore, it is better to force the CGA generation algorithm to stop after a certain time.

Aura and Roe [10] explained the possibility to select the Hash Extension parameters automatically instead of manual configuration to get more practical algorithm. But this idea does not standardize for CGA. Also, based on our knowledge, there is no implementation of this idea for CGA addresses. Therefore, we decide to modify the standard CGA and implement it based on stopping time.

III. MODIFICATIONS TO THE STANDARD CGA

For simplicity, usability, and practical requirements, it is better to determine Sec value in an automatic or indirect way based on tangible factor such as time. This section discusses the modified CGA algorithm with stopping time condition.

A. Stopping Time Condition for CGA

To guarantee that CGA generation process terminates after a certain time, the modified CGA algorithm takes the time as an input to determine the termination time. If this time threshold exceeds, the CGA generation algorithm stops. The algorithm keeps track of the best discovered value which has the highest number of zeros found in leftmost bits of Hash2. The extension length has to be rounded down to the nearest multiple integer of "16" or "8" to determine the security level for the rest of the CGA generation and verification algorithm.

It is better to set the security parameter (Sec) automatically based on a termination time rather than configuring it by the address generator (owner) for the following reasons:

- It is unreasonable to ask the user to understand the details of CGA algorithm to select the proper Sec value. But, it is possible to offer the user the possibility to select the value if she/he knows the details of the CGA algorithm.
- It is difficult to determine the proper Sec value, because Sec value has an exponentially scale effect on both the security and the computational cost.
- It depends on the time that the node has for configuring its address and on the application requirements. In mobile communication, the node should get its address within a certain time to achieve the handover.
- Not all devices have equal CPU speeds. Especially, mobile and embedded computers are likely to be much slower than a desktop workstation. Even if the user knows the details of the CGA algorithm, it is hard to select the practical Sec value for specific device. If the user has the possibility to select proper parameters, it is good to provide her/him with, at least, rough estimation about the expected time for each specific Sec value based on his device specifications.
- Preconfigured Sec value may compromise the security level. Setting the Sec to a fixed value (i.e., Sec = 0 or 1) may reduce the security of the resulting CGA over a period of time. The increase of computing power should be corresponding with the increment of Sec value to maintain the same level of security against brute-force attacks. Therefore, Setting Sec automatically based on a termination time offers an automatic adjust of Sec value based on the CPU speed. Faster CPU can achieve better security level within the same time. Accordingly, for the same termination time, the security level increases over the time by increasing the CPU speed. For the future devices with more powerful CPUs, the Hash Extension increases automatically.

The stopping time can be determined based on the maximum time for address generation. In fact, the maximum tolerable CGA address generation time depends on several factors. It depends on the device computing power, the particular application requirements, and other factors such as how long the user is willing to wait for CGA generation. Therefore, it is needed to select the proper termination time to get a feasible Sec value for CGA generation. The CPU speed of the address generator device can be used as an indication to the approximate estimation of the required CGA generates time [10]. In this manner, the Sec value can be selected or can be set indirectly to match the current available CPU speed.

The Sec is treated exactly in the same way as in the standard CGA similar that there is no time parameter in used. The method of determining the Sec value is independent of the mechanism for communicating it. Also, the CGA verification process remains the same as it is in standard CGA.

B. Selection the Granularity for Hash Extension Condition

The time-based stopping condition may waste the CPU resources because the multiple factor "16" is relatively large. CGA generator computes multiple Hash2 values during a time period defined by the time parameter input, and the output value is the one which has the greatest number of zeros bits. Since the number of zeros is expressed in 16×Sec, the most secure value is determined by selecting Hash2 value that has the greatest number of zeros bits rounded down to the nearest integer multiple of "16". Most likely the final value of Sec is reached early on the generation process, and the rest of the brute-force search will not find any better results (Sec + 1).

Smaller multiple factor is more suitable for TB-CGA. The multiple factor "16" was chosen to increase the maximum length of the Hash Extension up to 112 bits, but the benefit of this is questionable for the current CPU speeds. Therefore, for the TB-CGA, selecting the factor "8" instead of "16" is more reasonable for these reasons:

- It is more useful to round down to the nearest smaller multiple factor 8-bit instead of 16-bit to reduce the wasting time and achieve better security level. The chance to have "8" successive zeros is more than "16" successive zeros, especially for short stopping time parameter. Thus, having "8" zeros is better than round it to Sec value "0" in case the multiple factor is "16".
- Currently, Sec value "0" or "1" can be used in practical application. For Sec = 2, CGA address generation process may take several hours or days. So, having values in between (8 zeros or 24 zeros) is better than rounding down to the nearest integer of "16".

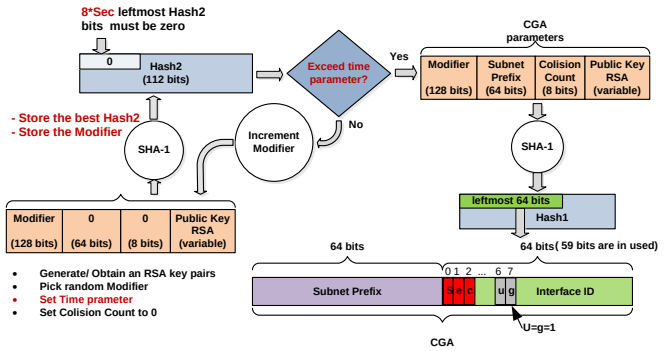

Figure 2. Time-Based CGA generation algorithm

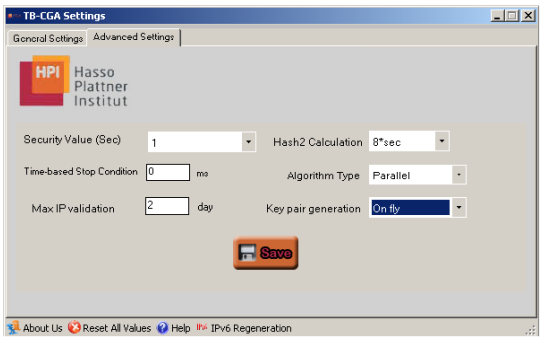

Figure 3. TB-CGA settings parameters

- The multiply factor "8" increases the maximum length of the Hash Extension up to 56 bits. Therefore, the total hash length will be between 59 up to 115 bits, which is enough for the current CPU speeds.

C. *Time-Based CGA (TB-CGA) Generation Algorithm*

Figure 2 shows a schematic of TB-CGA generation algorithm. Instead of Sec value, the time parameter is used as an input. If the time parameter has not been exceeded, increment the Modifier and compute a new Hash2 value. After generating each Hash2 value, the number of zero bits are counted and compared to the number of zero bits of a previous computed Hash2 value. During the brute-force search loop, Hash2 that matches the largest number of zeros in its leftmost bits is stored. Besides, the corresponding Modifier which results to the "best" Hash2 value is stored. When the time parameter is exceeded and the loop terminates the Modifier value that produces the highest found Sec value will be used for the remainder of the CGA address generation and verification.

IV. Experiment Setup And Implementation

We run our CGA implementation on guest Windows 7 operating system hosted by Virtualbox4.1.0 software. The settings of VirtualBox offer the flexibility to control the CPU execution capacity. For example, setting the execution capacity to 50% means a single virtual CPU can use up to 50% of a single host CPU. In our experiments, the hosted machine has 2.67 GHz CPU speed. Thus, the guest machine can use maximum up to 2.67 GHz.

Windows 7 guest runs WinSEND Analyzer implementation. WinSEND Analyzer is a light program for analyzing the CGA and SEND implementation for Windows families. It is used to analyze WinSEND implementation [11]. WinSEND Analyzer uses WinSEND main classes with some modifications to implementTB-CGA.

WinSEND Analyzer generates CGA based on selected parameters. WinSEND Analyzer Interface offers the flexibility for selecting the standard CGA or the TB-CGA (see Figure 3). Also, it offers the possibility to choose the desired CGA parameters. The user can select the RSA key size and determine how many times the CGA generation will be computed. If the standard CGA is used, the user can set the desired security level. In case the TB-CGA in use, the user sets

```
CGA parameter Data Structure
============================
Final Modifier: 9e8c313519756bf4ad5515159535f674
Subnet Prefix : 2007fe5aab8c7dc0
Collision Count : 00
Public Key: d48f5137175003313c013d377b6a2eb188c7ed371158d304
73088cd090c7a954a04e0584428564fdb4a42546bef73b6b8c474785e8c0
2e45dee98eae1c746e7c95186310471954d661a3a1842dee5f480c0dbf93
3b65227028ebcf8d4ec34f81f1569620640e17c0e6ee4d27256994fdcaa0
e7426f0276769fc8a166d5c182e7010001

The best founded Sec value (8*Sec): 1
Interface ID (CGA): 33e372e64fbe1439
Key size (RSA) = 1024-bit
Hash Algorithm: SHA-1
Hash1: 52e372e64fbe143951cb21008bfeb5f9b8e09c71
Hash2: 00545451ceabf1b52634d33039ce4ff225e1ee95

The stopping time: 200 milliseconds
The total number of iteration during the stopping time: 19166
Number of iteration to find best modifier :437
Time to find the best modifier: 140 milliseconds
```

Figure 4. A part of WinSEND analyzer output file that shows CGA parameter data structure and other statistical information for TB-CGA with multiple factor "8" for 200 milliseconds stopping time.

the CGA stopping time and the multiple factor for Hash2 condition (8 or 16). The final Sec value is the highest founded Sec value within the stopping time period. WinSEND Analyzer records and saves some measurements and statistics about the CGA generation process and writes it in an output text file. Figure 4 shows a part of WinSEND Analyzer output file which shows the CGA data structure and some other statistical information for TB-CGA with multiple factor "8" for stopping time 200 Milliseconds.

V. Performance Measurements

A. *Standard CGA Generation Time Measurements*

The CGA generation time measurements are taken by running WinSEND Analyzer on Windows 7 virtual machine over a range of CPU speeds started from 0.5 GHz up to 2.67 GHz. We choose this range to study the feasibility of using CGA for mobile devices. Now, the modern mobile devices have CPU speeds around 1 GHz. Some of these devices have Dual core 1.2 GHz. All the measurements are done for RSA key size equal to 1024-bit. The CGA address is generated 1000 times to have sufficient samples. The average (avrg.), the minimum (min.), and the maximum (max.) values of CGA generation times for Sec value "0" and "1" are recorded in Table I and Table II respectively. As it can be seen from Table I, the standard division of CGA generation time is high due to randomness of Hash2 and RSA key generation process. The

TABLE I. CGA GENERATION TIME WITH SEC = 0 FOR DIFFERENT CPU SPEEDS.

Virtual CPU speed (GHz)	Sec = 0 Number of samples (1000), RSA key size 1024-bit			
	CGA generation time (Milliseconds)			
	Avrg.	Min.	Max.	STD
0.5	251.59	30	940	119.46
1.0	168.25	40	650	74.65
1.5	137.79	40	510	68.13
2.0	101.74	10	500	63.34
2.5	97.29	10	350	50.95
2.67	93.41	3	360	50.48
Avrg.	**142.68**	---	---	---

TABLE II. CGA GENERATION TIME WITH SEC = 1 FOR DIFFERENT CPU SPEEDS.

Virtual CPU speed (GHz)	Sec = 1 Number of samples (1000), RSA key size 1024-bit				
	CGA generation time (Milliseconds)				Average number of tried Modifiers
	Avrg.	Min.	Max.	STD	
0.5	1047.94	80	6430	802.07	66604.48
1.0	691.82	60	3060	527.44	66920.81
1.5	570.09	50	3190	446.51	63870.86
2.0	512.67	50	2780	396.88	64050.86
2.5	439.64	4	2220	388.68	67728.37
2.67	401.99	10	2160	320.2	63155.97
Avrg.	**610.67**	---	---	---	**65388.56**

CGA generation time highly varies between the minimum and the maximum value.

Here, the CGA generation time is the total duration of the whole CGA generation process. It includes: the time for generating the RSA public/private keys, the time spent computing a Hash2 value that matches the condition 16×Sec-leftmost-bit of Hash2 are equal to zero, and the time for computing the interface identifier including Hash1 calculation. Beside the Duplicate Address Detection (DAD) check.

In Table II, the average number of tried modifiers indicates the average number of modifier values tried so far to find the Final Modifier in 1000 sample. Form Table II, the average number of iterations to find the Final Modifier over the range of CPU speeds from 0.5 to 2.67 GHz is 65388.56 iterations. This experimental result is close to the theoretical value which is ($2^{16\times1}$ = 65536 iterations). Therefore, we can say that 1000 CGA samples are sufficient to get relatively accurate approximation to the CGA measurements.

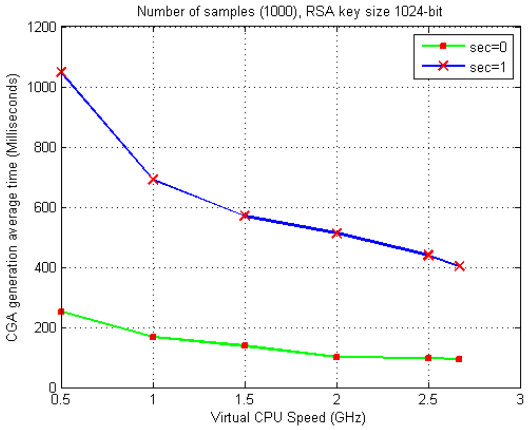

Figure 5. CGA generation average time for Sec = 0 and Sec = 1

As expected, the average CGA generation time for Sec= 1 is greater than the average CGA generation time for Sec =0. We found that increasing Sec value from "0" to "1" causes on average a CGA generation time to jump by a factor 469.01 Milliseconds. Figure 5 shows the CGA generation average time for both Sec = 0 and Sec = 1. From Figure 5, it can be seen that CGA generation average time decreases by increasing CPU speed, the trend curve for Sec=1 is:

$$T_{CGA}(S) = 714.5\ S^{-0.526} \quad \text{Milliseconds} \quad (1)$$

Where T_{CGA} is the average generation time and S is the CPU speed. Based on above equation, one can roughly predict the average required time to generate CGA address for Sec = 1 for different CPU speeds.

CGA with Sec value greater than "1" is unpractical with current CPU speed. Sec value "2" could be used in the next upcoming years. A test on unrepresentative set of 5 samples which carried out on 2.67 CPU speed gives on average 5923857 Milliseconds (1 hour and 39 minutes) CGA generation time. The average number of Hash2 computations is 1703473784 times. Still, Sec value "3" is not computationally feasible for the current CPU speeds since Sec value increases the CGA computation exponentially. CGA computation for Sec vale "3" will take on average more than 12 years if the CPU speed is 2.67 GHz.

B. Time-based CGA Generation Measurements

For TB-CGA, the algorithm searches for the largest Sec value during the termination time. The extension length is rounded down to the nearest multiple of 16-bit. The multiple factor "16" is quite large and cases a big jump between the two successive Hash Extensions. For Sec = 1, Hash2 must contains 16 zeros in the left most bits, while it is 32 zeros if Sec = 2. This large jump wastes CPU time of the address generator computing device because most probability the CGA generation algorithm will find the best Sec value early, and the rest of the brute-force search will not find any better Sec value.

Figure 6 shows the total number of founded Sec values over a range of stopping time started from 100 Milliseconds to 1500 Milliseconds for multiple factor "16". The experiment was

done on a computer with 2.6 GHz. For each stopping time value, the CGA is computed 1000 times. As you can see from Figure 6, most of the time only Sec values "0" are found. The percentage of founded Sec value "0" is equal to 96.25% while it is only 3.75% for Sec value "1", which means that most of the time the algorithm is busy for searching for higher Sec value than "1" but it rarely succeed. Therefore, it is better to reduce the multiple factor of successive zeros in the leftmost of Hash2 by using smaller multiple factor than "16".

Figure 7 shows the number Sec values for multiple factor "8". Sec value "1" (which means 8 successive zeros) dominates. It forms 80.05% of the founded Sec values. And only 12.53 % for Sec value "0". The algorithm successes to find one Sec value "3" which is means 24 zeros in Hash2. Definitely, having Sec value "1" in case of using the multiple factor "8" is better than having Sec value "0" in case the multiple factor "16" by rounding these "8" zeros down to nearest multiple integer "16". Sec value "1" in case of using the multiple factor "8" is more secure than Sec value "0" with multiple factor "16". Multiple factor "16" wastes the CPU computation without achieving better security level especially for short stopping time or for slow CPU speeds.

VI. CONCLUSION

CGA is an IPv6 bound to owner's public key. It provides an authentication mechanism in a decentralized way. The security parameter (Sec) has a great impact on the CGA generation time and it makes CGAs computationally costly. Fulfilling the second hash (Hash2) condition is the computationally expensive part of CGA generation.

In this paper we presented a practical and automatic way for selecting the Sec parameter for CGA generation algorithm. In this modified version, the time is taken as an input and then the Sec value is determined as an output of the brute-force search to satisfy Hash2 condition. The security level is determined automatically based on the computing device CPU power available for hash generation. Faster devices are able to find a better Sec value than slower ones for the same time. The communication of Sec value and CGA verification process remains the same as the standard CGA. For Time-base CGA, we recommend the use of multiple factor "8" instead of "16" in Hash Extension condition. The use of factor "8" reduces the steps between the successive Sec values and consequently reduces wasting computation to find better Sec vale.

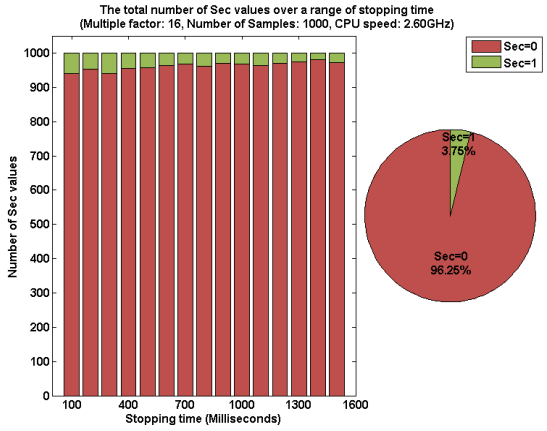

Figure 6. The number and the percentage of founded Sec values over a range of stopping time for multiple factor "16"

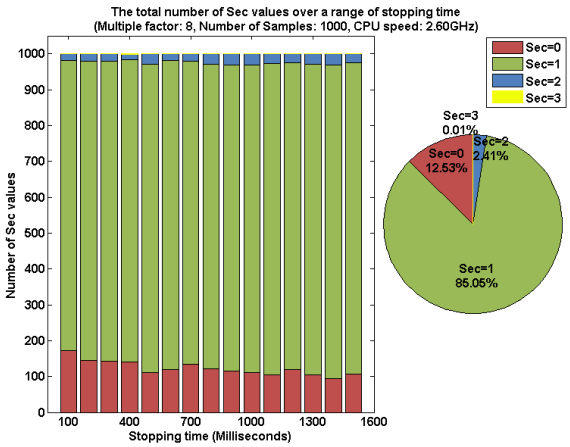

Figure 7. The number and the percentage of founded Sec values over a range of stopping time for multiple factor "8"

REFERENCES

[1] T. Aura, "Cryptographically Generated Address", RFC3972, Internet Engineering Task Force, March 2005, http://tools.ietf.org/html/rfc3972

[2] T. Aura, "Cryptographically generated addresses (CGA)", In Proceedings of the 6th Information Security Conference(ISC'03), Bristol, UK, LNCS, vol. 2851, pp. 29-43, 2003.

[3] G. O'Shea and M. Roe, "Child-proof authentication for MIPv6 (CAM)", ACM Computer Communications Review, vol. 31, no. 2. 2001.

[4] J. Arkko, J. Kempf, B. Zill and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971 (Proposed Standard), Internet Engineering Task Force, March 2005, http://tools.ietf.org/html/rfc3971.

[5] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, Internet Engineering Task Force, September 2007, http://tools.ietf.org/html/rfc4861.

[6] S. Thomson, T. Narten and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, Internet Engineering Task Force, September 2007, http://tools.ietf.org/html/rfc4862.

[7] P. Nikander, J. Kempf and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756 (Informational), Internet Engineering Task Force, May 2004, http://tools.ietf.org/html/rfc3756.

[8] J. Arkko, C. Vogt and W. Haddad, "Enhanced route optimization for mobile IPv6", RFC 4866, Internet Engineering Task Force, May 2007, http://tools.ietf.org/html/rfc4866.

[9] T. Aura, M. Roe, "Designing the Mobile IPv6 Security Protocol", Annals of telecommunications, special issue on Network and information systems security, volume 61 number 3-4, March-April 2006.

[10] T. Aura and M. Roe, "Strengthening Short Hash Values" http://research.microsoft.com/en-us/um/people/tuomaura/misc/aura-roe-submission.pdf.

[11] H. Rafiee, A. AlSa'deh, and Ch. Meinel, "WinSEND: Windows SEcure Neighbor Discovery", 4th International Conference on Security of Information and Networks (SIN 2011), 14-19 November 2011, Sydney, Australia, pp.: 243-246, November 2011.