IETF Securing Neighbor Discovery WG
INTERNET DRAFT
Expires August 2003
February 2003

Tuomas Aura
Microsoft Research

Cryptographically Generated Addresses (CGA)
draft-aura-cga-00.txt

Status of This Memo

Abstract

Cryptographically generated addresses (CGA) are IPv6 addresses where the interface identifier is generated by hashing the address owner's public key. The address owner can then use the corresponding private key to assert address ownership and to sign messages sent from the address without any additional security infrastructure. This document describes a generic CGA format that can be used in multiple applications.

Table of Contents

1. Introduction

This document specifies how to create IPv6 addresses from the cryptographic hash of a public key (and auxiliary parameters). Public-key signatures can then be used for authenticating messages from the address owner. The main advantage of the CGA-based authentication is that additional security infrastructure, such as a PKI or TTP, is not needed. Potential applications include Mobile IPv6 binding update authentication (e.g. as a part of the CAM protocol [OR01]), proof of address ownership in secure neighbor discovery and duplicate address detection [AAK+02], and key exchange for opportunistic IPSec encryption and authentication.

The address format defined in this document differs from previous proposals [OR01][Nik01][MC02] at least in the following respects:

(1) Two hash values are computed instead of one. The first hash value (Hash1) is used to produce the Interface Identifier (i.e. rightmost 64 bits) of the address. The purpose of the second hash (Hash2) is to artificially increase that computational complexity of generating new addresses and, consequently, the cost of brute-force attacks. This allows the address owner to select levels of security above the 62-bit limit of CAM.

(2) The Routing Prefix (i.e. leftmost 64 bits) of the address is included in the first hash input, which makes some brute-force attacks against global-scope addresses more expensive because the attacker must do a separate brute-force search for each address prefix. However, we take care not to make mobility more expensive for the address owner. When the Routing Prefix changes, the second hash value can be reused, thus avoiding the expensive brute-force part of address generation.

(3) The input to both hash functions is formatted as (parts of) a self-signed X.509 v3 certificate. This has several advantages. First, a self-signed certificate is a standard format for storing and transferring public keys in Internet protocols. Second, the signature on the certificate proves that the public-key owner wants to use the IPv6 address. Third, future protocols may bind arbitrary security-critical information (other than the address owner's public key) to the IPv6 address by defining a new type of certificate extension for that purpose. Fourth, the use of X.509 v3 certificates makes it easy to use CGA-based and PKI-based address authentication side by side in the same protocols. Some protocols, however, may need to save octets and transfer only the public key and other absolutely necessary

parameters, rather than a full self-signed certificate. An optimized parameter format is defined for this purpose.

In order to verify the address owner's signatures, one needs to have the address itself and the associated self-signed certificate, which contains the public key. The address format and certificate format are defined in Sections 2 and 3. The detailed algorithms for generating addresses and for verifying them are given in Sections 4 and 5. Finally, Section 6 discusses security of the technique.

2. The CGA Address Format

The leftmost 64 bits of the 128-bit IPv6 Address form the Routing Prefix. The rightmost 64 bits of the address are called the Interface Identifier.

Cryptographically generated addresses also have a security parameter (Sec), which determines the level of security. The security parameter is a 3-bit unsigned integer encoded in the three rightmost bits of the 128-bit IPv6 address.

```
                        Sec = Address & 7
```

The address is associated with a self-signed X.509 v3 certificate, which contains the address owner's public key. Two hash values Hash1 and Hash2 are computed from parts of the certificate. The format of the certificate and the inputs to the hash functions are defined in Section 3.

A cryptographically generated address (CGA) is defined as an IPv6 address where the 12*Sec leftmost bits of the second hash value Hash2 are zero, and the rightmost 64 bits of the first hash value Hash1 equal the Interface Identifier of the address. The three rightmost bits of the address, which encode the security parameter Sec, and the universal and group bits are ignored in the comparison. The latter two bits must both be one. [Alternatively, we can stick with U=1, G=0. That would make it difficult to use CGA-based authentication side by side with weaker protocols.]

The above definition can be stated in terms of the following three bit masks (Mask1, Mask2, Mask3):

```
  Mask1 = 0x00000000000000000000000000000000  if Sec=0,
          0xfff00000000000000000000000000000  if Sec=1,
          0xffffff00000000000000000000000000  if Sec=2,
          0xfffffffff00000000000000000000000  if Sec=3
          0xffffffffffff00000000000000000000  if Sec=4,
          0xfffffffffffffff00000000000000000  if Sec=5,
```

```
             0xffffffffffffffffff00000000000000  if Sec=6, and
             0xfffffffffffffffffffff00000000000  if Sec=7

     Mask2 = 0x00000000000000000300000000000000

     Mask3 = 0x0000000000000000fffffffffffffff8
```

A cryptographically generated address is an IPv6 address for which the following are true:

```
            (Hash1 & Mask3) | Mask2  ==  Address & Mask3
                        Hash2 & Mask1 == 0
```

3. The CGA Certificate and the Hash Values

Each CGA address is associated with a self-signed X.509 v3 certificate [HFPS02][ITU97]. The subjectPublicKeyInfo data value in the certificate is the address owner's public key. A certificate extension contains the following parameters: a 12-octet Modifier, the 8-octet Routing Prefix of the address, and Collision Count, which can get values 0, 1 and 2. The extnID field in the extension has the following value:

```
             cgaExtnID = { 1 3 6 1 4 1 311 TBD }
```

The critical field in the extension MAY be set to false or true, depending on whether the certificate has other uses than CGA-based authentication. The extnValue field in the extension contains a DER-encoded data value of the following type:

```
             CGAParameters ::= SEQUENCE {
               modifier       OCTET STRING (SIZE 12),
               routingPrefix  OCTET STRING (SIZE 8),
               collisionCount INTEGER (0..2) }
```

Two 128-bit hash values Hash1 and Hash2 are computed with the MD5 algorithm from parts of the certificate. The input to Hash1 is the concatenation of the DER-encoded subjectPublicKeyInfo and CGAParameters data values. The input to Hash2 is the concatenation of the DER-encoded subjectPublicKeyInfo and modifier data values.

As an alternative to the certificate, an optimized parameter format MAY be used. The optimized format is simply the concatenation of the subjectPublicKeyInfo and CGAParameters data values. Security protocols that use CGA addresses MUST specify whether they use the certificate format or the optimized parameter format. The same address MAY be used in both types of protocols.

Note 1: The DER encoding of the CGAParameters data value is 29 octets long and has the following (hexadecimal) format: 30 1d 04 0c xx xx xx xx xx xx xx xx xx xx xx xx 04 08 yy yy yy yy yy yy yy yy 02 01 zz, where xx..xx is the Modifier, yy..yy is the Routing Prefix, and zz is the Collision Count. All the 29 octets are included in the input to Hash1. Only the 12 octets xx..xx are included in the input to Hash2.

4. CGA Generation

The process of generating a new CGA takes three input values: a 64-bit Routing Prefix, the Public Key of the address owner, and the security parameter Sec, which is an unsigned 3-bit integer. The result is a new CGA and the associated self-signed certificate (as defined in Sections 2-3). The cost of generating a new CGA depends on the security parameter Sec, which gets values from 0 to 7.

If Sec=0, a CGA can be generated from the hash input with the following steps:

(1) DER-encode the address owner's public key as an ASN.1 structure of the type SubjectPublicKeyInfo.

(2) Create an ASN.1 structure of type CGAParameters. Set the modifier data value to 12 zero octets. Set the routingPrefix data value to be the Routing Prefix. Set the collisionCount data value to zero. DER-encode the CGAParameters data value.

(3) Concatenate the DER-encoded SubjectPublicKeyInfo and CGAParameters data values. Execute the MD5 algorithm on the concatenation. The result is Hash1.

(4) Concatenate the 64-bit Routing Prefix and the rightmost 64 bits of Hash1 to form a 128-bit IPv6 address.

(5) Set the group and universal bits in the address both to 1 and the three rightmost bits of the address all to 0.

(6) If an address collision is detected, increment the collisionCount data value in the DER-encoded CGAParameters data value and go back to step (3). However, after three collisions, stop and report the error.

(7) Create and sign a self-signed X.509 v3 certificate using the SubjectPublicKeyInfo data item created in step (1). Include in the certificate an extension where the extnID has the value cgaExtnID, critical has the value false or true, and the extnValue contains the encoded CGAParameters data value

from step (2). The certificate MAY contain other fields and extensions.

If the generated CGA will be used only in protocols that that use the optimized parameter format, step (9) MAY be skipped.

Nodes MAY set the security parameter Sec to zero and implement only the above procedure, which is deterministic and relatively fast. However, it is RECOMMENDED that implementations support the generation of addresses with higher Sec values. For any Sec value from 0 to 7, a CGA can be created as follows:

(1) DER-encode the address owner's public key as an ASN.1 structure of the type SubjectPublicKeyInfo.

(2) Create an ASN.1 structure of type CGAParameters. Set the modifier data value to 12 random octets. Set the routingPrefix data value to be the Routing Prefix. Set the collisionCount data value to zero. DER-encode the CGAParameters data value.

(3) Concatenate the DER-encoded SubjectPublicKeyInfo and modifier data values. Execute the MD5 algorithm on the concatenation. The result is Hash2.

(4) Compare the 12*Sec leftmost bits of Hash2 with zero. If they are all zero (or if Sec=0), continue with the step (5). Otherwise, increment the modifier data value (as if the content octets of the modifier were a 96-bit integer) and go back to step (3).

(5) Concatenate the DER-encoded SubjectPublicKeyInfo and CGAParameters data values. Execute the MD5 algorithm on the concatenation. The result is Hash1.

(6) Concatenate the 64-bit Routing Prefix and the rightmost 64 bits of Hash1 to form a 128-bit IPv6 address.

(7) Set the group and universal bits in the address both to 1 and the three rightmost bits of the address to the value Sec.

(8) If an address collision is detected, increment the collisionCount data value in the encoded CGAParameters data value and go back to step (5). However, after three collisions, stop and report the error.

(9) Create and sign a self-signed X.509 v3 certificate using the SubjectPublicKeyInfo data item created in step (1). Include

in the certificate an extension where the extnID has the value cgaExtnID, critical has the value true or false, and the extnValue contains the encoded CGAParameters data value from step (2). The certificate MAY contain other fields and extensions.

If the generated CGA will be used only in protocols that that use the optimized parameter format, step (9) MAY be skipped.

Note 1: The initial value of Modifier in step (1) and the method of modifying it in step (4) MAY be chosen arbitrarily. In order to avoid trying the same Modifier values repeatedly, it is RECOMMENDED that the initial value is chosen randomly. The quality of the random number generator is not important as long as the same values are not repeated frequently. The RECOMMENDED way to modify Modifier is to increment the content octets of the modifier data value as if they were an unsigned 96-bit integer. (Octet order can be chosen arbitrarily and overflows can be ignored.)

Note 2: For security parameter values greater than 0, this second algorithm is not guaranteed to terminate after a certain number of iterations. The brute-force search in steps (3)-(4) takes on the average approximately 2^(12*Sec) iterations to complete.

Note 3: If the Routing Prefix of the address changes but the address owner's public key does not, the old value of Modifier can be used and it is unnecessary to repeat the brute-force search.

5. CGA Verification

CGA verification takes two inputs: an IPv6 address and a self-signed X.509 v3 certificate. In protocols where saving octets is essential, the certificate MAY be replaced by the optimized parameter format (i.e. a concatenation of the DER-encoded SubjectPublicKeyInfo and CGAParameters data items). The verification either succeeds or fails. If the verification succeeds, the verifier knows that the certificate contains the public key of the address owner. The verifier can then use the public key to authenticate signed messages from the address owner or to exchange a session key with the address owner.

The CGA is verified with the following steps:

(1) Compare the group and universal bits in the address to one. If either bit is zero, the address is a non-CGA address and no verification can be done.

(2) Read the security parameter Sec from the three rightmost bits of the address. (Sec is an unsigned 3-bit integer.)

(3) Find the encoded subjectPublicKeyInfo data value in the certificate.

(4) Find and decode a certificate extension with extnID value equal to cgaExtnID. Decode the content octets of the corresponding extnValue data value, which have the type CGAParameters. Check that the collisionCount value is 0, 1 or 2. The CGA verification fails if the extension does not exist, if decoding of the extension fails, or if the collisionCount value is out of range.

(5) Check that the routingPrefix data value is equal to the Routing Prefix (i.e. leftmost 64 bits) of the address.

(6) Concatenate the encoded SubjectPublicKeyInfo and CGAParameters data values. Execute the MD5 algorithm on the concatenation. The result is Hash1.

(7) Take the rightmost 64 bits of Hash1 output and compare them with the Interface Identifier (i.e. the rightmost 64 bits) of the address. Differences in the group and universal bits and in the three rightmost bits are ignored. If the 64-bit values differ (other than in the five ignored bits), the CGA verification fails.

(8) Concatenate the encoded SubjectPublicKeyInfo and modifier data values. Execute the MD5 algorithm on the concatenation. The result is Hash2.

(9) Compare the 12*Sec leftmost bits of Hash2 with zero. If any one of these is non-zero, CGA verification fails. If Sec=0, verification never fails at this step.

(10)
Verify the signature on the self-signed certificate. If the signature is invalid, the GCA verification fails. Otherwise, the CGA verification succeeds.

All nodes that verify CGAs MUST be able to process all security parameter values Sec = 0, 1, 2, 3, 4, 5, 6, 7. The verification procedure is relatively fast and always requires a constant amount of computation.

In protocols where the optimized parameter format is used instead of the certificate format, the signature verification in step (10) is skipped.

Note 1: The values of Modifier and Collision Count are ignored in the CGA verification procedure, except for checking that Collision

Count is in the allowed range in step (3) and for including them into the appropriate hash inputs.

6. Security Considerations

The purpose of CGA addresses is to prevent stealing and spoofing of IPv6 addresses. The public key of the address owner is bound cryptographically to the address. The address owner can use the corresponding private key to assert its ownership of the address and to sign messages sent from the address.

It is important to understand that that attacker can create a new address from an arbitrary routing prefix and its own public key. What the attacker cannot do is to impersonate somebody else's address. This is because the attacker would have to find a collision of the cryptographic hash value Hash1. (The property of the hash function needed here is called second pre-image resistance.)

The signature on the self-signed certificate proves that the owner of the public key wants to be associated with the address. The signature also binds other certificate fields to the address. Protocols that use CGAs but need to bind additional information (other than the public key) to the address may define new certificate extensions for this purpose.

Some CGA applications may need to sign individual IPv6 packets. A CGA-signed packet will have the CGA address as its source address, and it will have to contain the associated certificate, a payload, and a signature. There is the problem that the packet size may exceed the Ethernet MTU. In that case, the optimized parameter format, rather than the full certificate, can be sent to the verifier. (In fact, the full certificate need not be created if it is not needed for other protocols.) Protocols that use this optimization obviously cannot require verification of the signature on the certificate. These protocols should include the source address of the packet in the signed message in order to prove that the public-key owner wants to use the address. For simplicity, the signature on the self-signed certificate MUST always be verified if a certificate is available to the verifier.

As computers become faster, the 64 bits of the Interface Identifier will not be sufficient to prevent attackers from searching for hash collisions. It helps somewhat that we include the routing prefix of the address in the hash input. This prevents the attacker from using a single pre-computed database to attack addresses with different routing prefixes. The attacker needs to create a separate database for each routing prefix. Link-local addresses are,

however, left vulnerable because the same routing prefix is used by all IPv6 nodes.

In the long term, some kind of hash extension technique must be used to counter the effect of faster computers. Otherwise, the CGA technology could become outdated after 5-20 years. The idea in this document is to increase the cost of both address generation and brute-force attacks by the same parameterized factor while keeping the cost of address use and verification constant. This provides protection also for link-local addresses.

For this purpose, the input to a second hash function Hash2 is modified by varying the value of Modifier until the leftmost 12*Sec bits of Hash2 are zero. This increases the cost of address generation approximately by a factor of 2^(12*Sec). It also increases the cost of brute-force attacks by the same factor. That is, the cost of creating a certificate that binds the attacker's public key with somebody else's address is increased approximately by a factor of 2^(12*Sec). The address owner may choose the security parameter Sec depending of its own computational capacity and the expected lifetime of the address. Currently, Sec values between 0 and 2 are sufficient for most IPv6 nodes. As computers become faster, higher Sec values will slowly become useful.

Theoretically, if a typical attacker is able to tap into N local networks at the same time, an attack against link-local addresses is N times as efficient as an attack against addresses of a specific network. This effect can be countered by using log2(N) bits longer hash extensions for link-local addresses than for global-scope addresses. (Incrementing Sec by one causes a 12-bit increase in the length of the hash extension.)

In order to make it possible for mobile nodes whose routing prefix changes frequently to use Sec values greater than 0, we have decided not to include the routing prefix in the input of Hash2. The result is weaker than if the routing prefix were included in the input of both hashes. On the other hand, our scheme is at least as strong as using the hash extension technique without including the routing prefix in either hash. It is also at least as strong as not using the hash extension but including the routing prefix. This trade-off was made because mobile nodes frequently move to insecure networks where they are at the risk of denial-of-service attacks, for example, during the duplicate address detection procedure.

In most applications of CGA, the goal is prevent denial-of-service attacks. Therefore, it is usually sensible to start by using a low Sec value and to replace addresses with stronger ones only when denial-of-service attacks based on brute-force search become a significant problem. On the other hand, if CGA is used as a part of

a strong authentication or secrecy mechanisms (e.g. CGA authentication plus Secure DNS), then it may be necessary to start with higher Sec values. (Fortunately, link-local addresses are not used in the latter kind of applications.)

Collision Count is used to modify the input to Hash1 if there is an address collision. It is important not to allow Collision Count values higher than 2. First, it is extremely unlikely that three collisions would occur and the reason is certain to be either a configuration or implementation error or a denial-of-service attack. (These DoS attacks can be prevented if the IPv6 neighbor discovery messages are authenticated with CGA addresses.) Second, an attacker who is doing a brute-force search to match a given CGA address can try all different values of Collision Count without repeating the brute-force search for Modifier. Thus, the more different values are allowed for Collision Count, the less effective the hash-extension technique is in preventing brute-force attacks.

It is important to understand that when a CGA address is used to authenticate messages from an IPv6 node, the receiver of the message must know the exact IPv6 address. In network layer signaling, such as duplicate address detection and Mobile IPv6 binding updates, the IPv6 address is the natural identifier for a network node. In other applications, such as opportunistic IPSec, it is possible to get around the protection by tricking the receiver into accept the wrong IPv6 address, e.g. by DNS spoofing, unless the address resolution is protected by at least equally strong mechanisms.

Finally, CGA-based authentication has some implications on privacy. The CGA addresses can be randomized by choosing a random initial value for Modifier and by generating a new address at desired intervals. If the node reveals the associated certificate or public key to its correspondents, it should also replace the public key at the same time as the address. This gives the same level of protection as the IPv6 address privacy extensions [ND01]. However, the cost of public-key and address generation may imply less frequent address changes.

Acknowledgments

Many of the ideas in this draft were influenced by Michael Roe, Christian Huitema and Pekka Nikander.

Intellectual Property Statement

The author believes that there are several patents or patent applications that cover parts of this specification.

References

[AAK+02] Jari Arkko, Tuomas Aura, James Kempf, Vesa-Matti M ntyl , Pekka Nikander, and Michael Roe. Securing IPv6 neighbor discovery and router discovery. In Proc. 2002 ACM Workshop on Wireless Security (WiSe), pages 77-86, Atlanta, GA USA, September 2002. ACM Press.

[Eas99] Donald Eastlake. Domain name system security extensions. RFC 2535, IETF Network Working Group, March 1999.

[HD98] Robert M. Hinden and Stephen E. Deering. IP version 6 addressing architecture. RFC 2373, IETF Network Working Group, July 1998.

[HFPS02] Russell Housley, Warwick Ford, Tim Polk, and David Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 3280, IETF Network Working Group, April 2002.

[ITU97] International Telecommunication Union. ITU-T recommendation X.509 (1997 E): Information technology - open systems interconnection - the directory: Authentication framework, June 1997.

[ITU02] International Telecommunication Union. ITU-T recommendation X.690, information technology -- ASN.1 encoding rules: Specification of basic encoding rules (BER), canonical encoding rules (CER) and distinguished encoding rules (DER), July 2002. Also appeared as ISO/IEC International Standard 8825-1.

[JB99] Stephen Kent and Randall Atkinson. Security architecture for the Internet Protocol. RFC 2401, IETF Network Working Group, November 1998.

[MC02] Gabriel Montenegro and Claude Castelluccia. Statistically unique and cryptographically verifiable identifiers and addresses. In Proc. ISOC Symposium on Network and Distributed System Security (NDSS 2002), San Diego, February 2003.

[Mos01] Robert Moskowitz. Host identity payload and protocol. Internet-Draft draft-ietf-moskowitz-hip-05.txt, October 2001. Work in progress.

[ND01] Thomas Narten and Richard Draves. Privacy extensions for stateless address autoconfiguration in IPv6. RFC 3041, IETF Network Working Group, January 2001.

[NN98] Thomas Narten, Erik Nordmark, and William Allen Simpson. Neighbor discovery for IP version 6 (IPv6). RFC 2461, IETF Network Working Group, December 1998.

[Nik01] Pekka Nikander. A scaleable architecture for IPv6 address ownership. Internet-draft, March 2001. Work in Progress.

[OR01] Greg O'Shea and Michael Roe. Child-proof authentication for MIPv6 (CAM). ACM Computer Communications Review, 31(2), April 2001.

[TN98] Susan Thomson and Thomas Narten. IPv6 stateless address autoconfiguration. RFC 2462, IETF Network Working Group, December 1998.

Authors' Addresses

Tuomas Aura
Microsoft Research
7 J J Thomson Avenue
Cambridge, CB3 0FB
United Kingdom

Phone: +44 1223 479708
Email: tuomaura@microsoft.com