

Cryptographically Generated Addresses (CGA)

Tuomas Aura

Microsoft Research

Roger Needham Building, 7 JJ Thomson Avenue, Cambridge CB3 0FB, UK
tuomaura@microsoft.com

Abstract. Cryptographically generated addresses (CGA) are IPv6 addresses where some address bits are generated by hashing the address owner's public key. The address owner uses the corresponding private key to assert address ownership and to sign messages sent from the address without a PKI or other security infrastructure. This paper describes a generic CGA format that can be used in multiple applications. Our focus is on removing weaknesses of earlier proposals and on the ease of implementation. A major contribution of this paper is a hash extension technique that increases the effective hash length beyond the 64-bit limit of earlier proposals.

1. Introduction

Cryptographically generated addresses (CGA) are IPv6 addresses where some of the address bits, usually the 64-bit interface identifier, are created from a cryptographic hash of the address owner's public key. The address owner uses the corresponding private key to sign messages sent from the address or to assert its ownership of the address. Anyone can verify which public key matches the address. The only way to circumvent this authentication is to find another public key that produces the same hash, i.e. to break the second pre-image resistance of the cryptographic hash function.

The main advantage is that no third parties or additional infrastructure, such as a public-key infrastructure (PKI), are needed. Any IPv6 node capable of basic cryptographic operations can generate a CGA address locally, and only the address and the public key are needed for verifying the signatures. CGA-based authentication is particularly useful in securing IP-layer signaling protocols, such as neighbor discovery, where IPv6 addresses are the primary identifiers for the protocol participants. This paper describes a generic format for cryptographically generated addresses (CGA) that can be used in many such protocols.

We have solved the main security weakness of earlier CGA proposals by effectively removing the 64-bit limit on the hash length. The hash extension is important because otherwise exponential growth of computing power could enable an attacker to break CGA-based authentication and, thus, protocols that depend on it in the foreseeable future. We achieve the improvement by artificially increasing both the cost of generating a new CGA address and the cost of a brute-force attack while keeping the cost of CGA-based authentication constant.

Potential applications for CGA-based authentication include Mobile IPv6 binding update authentication, proof of address ownership in secure neighbor discovery and duplicate address detection, and key exchange for opportunistic IPsec encryption and authentication.

Sections 2 and 3 survey related work and give some preliminary information. The CGA address and parameter formats are defined in Section 4. Detailed algorithms for generating addresses and for verifying them are given in Section 5. Section 6 explains our design decisions and Section 7 overviews promising applications on which there is ongoing work.

2. Related Work

Cryptographically generated addresses first appeared in the CAM protocol for Mobile IPv6 binding update authentication by O'Shea and Roe [OR01]. Nikander [Nik01] suggested using the same technique for proofs of address ownership. Montenegro and Castelluccio [MC02] worked on similar proposal for Mobile IPv6, called statistically unique and cryptographically verifiable addresses (SUCV). All these addresses are self-certifying in the same sense as the self-certifying file names by Mazières et al. [MKKW99]. Our specification differs from the earlier CGA proposals by defining a general-purpose address format rather than focusing on specific applications. We also remove the main security weakness of the earlier techniques, the 64-bit limit on hash length.

An alternative to CGA are different types of public-key infrastructures (PKI). In X.509, certificates issued by a hierarchy of certification authorities (CA) bind names to public keys. An X.509 certificate can be used to bind an IP address to the address owner's public key because the address can be seen as a name for its owner. KeyNote [BFIK99a] and SPKI [EFL+99] support qualified trust relations and authorization in addition to naming. These systems could be used to authorize a public key to send messages from an IP address. Unlike CGA, which depends on the 128-bit IPv6 addresses, the PKI-based security solutions work equally well for IPv4.

Identity-based cryptography [Sha84] is an alternative to certificates. In identity-based signature schemes any identifier can be used as a public key. A trusted authority generates and distributes the corresponding private key to the owner of the identifier. Address-based keys (ABK) [ODG+02] are an application of this technique to IP address authentication. Any IP address can be used as the public key. ABK has the same infrastructure requirements and security properties as a certificate-based PKI: a trusted party is needed for distributing the private keys. The performance of ABK is better, however, because the verifier of ABK-based authentication does not need to obtain or verify a certificate.

The biggest advantage of CGA-based authentication compared to a PKI or ABK is that it does not require a trusted authority, pre-established trust relationships or other security infrastructure.

The HIP architecture (see Moskowitz [Mos01] and Nikander et al. [NYW03]) goes one step further than CGA and suggests using a hash of the public key as the primary identifier for IP nodes. In the host identity payload (HIP) protocol, a new protocol

```

Sec = (interface identifier & 0xe000000000000000) >> 61
Mask1 (112 bits) = -1 << (16*Sec)
Mask2 (64 bits) = 0x1cfffffffffffffff

A cryptographically generated address is an IPv6 address for which the following
two equations hold:
    Hash1 & Mask2 == interface identifier & Mask2
    Hash2 & Mask1 == 0x00000000000000000000000000000000

```

Fig. 1. The definition of a CGA address using bit masks

layer between the transport and network layers maps the hash-based identifiers to routable IP addresses and provides authentication. While this is a promising direction for the long term, the changes required to the IP protocol stack are much more drastic than for CGA.

CGA uses the IP-layer addressing architecture to bootstrap security. Another piece of infrastructure always available in the IP layer is the best-effort routing architecture. In some cases, the partial reliability of routing can be used for authentication. For example, many authentication protocols (e.g. Photuris [KS99]) start with a cookie exchange that can be seen as a weak form of authentication. A Mobile IPv6 binding update authentication protocol [Aur02] based on the same idea has been adopted into the IETF Mobile IPv6 specification [JPA03]. A CGA-based protocol was a strong alternative but the relatively weak security requirements favored the less demanding routing-based approach in that case.

3. Preliminaries

An IPv6 address 0 is 128 bits long. It is divided into two parts. The leftmost 64 bits, the *subnet prefix*, is used for routing IP packets across the Internet to the destination network. The rightmost 64 bits, the *interface identifier*, identifies an individual node within a local network. The interface identifiers may be chosen in an arbitrary way, e.g. randomly, as long as no two nodes on the same network share the same value.

Two bits of the interface identifier have a special semantics. The 7th bit from the left is the Universal/Local bit or “u” bit. It is usually set to 1 to mean that the interface identifier is configured from an EUI-64 identifier from the interface hardware and, thus, is globally unique. The 8th bit from the left is the Individual/Group or “g” bit, which is set to 1 for multicast addresses. The bit combination $u=1, g=1$ is currently unused because a multicast address cannot be globally unique. We suggest allocating this bit combination for CGA addresses.

4. CGA Specification

This section defines the format of a CGA address (Section 4.1) and the associated parameters and hash values (Section 4.2). The reader should note that the CGA format is being specified by the IETF and while the our description mostly follows the latest draft specification at the time of writing, the details are subject to change.

4.1 The CGA Address Format

Cryptographically generated addresses have a *security parameter* (Sec), which determines the level of security. The security parameter is a 3-bit unsigned integer. It is encoded in the three leftmost bits of the 64-bit interface identifier. The parameter allows the address owner to increase cost of address generation and, thus, the cost of brute-force attacks against the address. Incrementing Sec by one adds 16 bits to the length of the hash that the attacker must break. Since the parameter value is encoded into the address bits, an attacker cannot change its value without also changing the address.

The address is associated with a public key and auxiliary parameters, from which two hash values Hash1 and Hash2 are computed. The format of the parameters and the inputs to the hash functions are defined in Section 4.2.

A cryptographically generated address (CGA) is defined as an IPv6 address where the $16 * \text{Sec}$ 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 leftmost bits of the interface identifier, which encode the security parameter Sec, and the “u” and “g” bits are ignored in the comparison. We suggest setting the latter two bits both to 1 but that is currently not required.

The above definition can be stated in terms of bit masks as shown in Figure 1. Thus, the specification is straightforward to implement.

As defined above, the first hash value Hash1 is used to create 59 bits of the interface identifier in the address. The purpose of the second hash value is to increase artificially the cost of brute-force attacks where the attacker varies the hash input in order to match its own public key with somebody else's address. In addition to matching the 59 address bits with Hash1, an attacker must match the $16 * \text{Sec}$ zero bits with Hash2. This technique, called *hash extension*, effectively increases the hash length beyond the 64-bit boundary of earlier proposals. The resulting level of security against brute-force attacks is equal to a hash length of $59 + 16 * \text{Sec}$ bits.

In order to generate an address with $\text{Sec} > 0$, the address owner must also perform a brute-force search. However, the address owner gets the 59 address bits free from Hash1 and only needs to match the $16 * \text{Sec}$ zero bits with Hash2. It is important to note that once the address has been created, the cost of using and verifying a CGA address does not depend on the value of Sec.

4.2 The CGA Parameters and Hash Values

As mentioned above, every CGA address is associated with a public key and auxiliary parameters. The address owner's public key is formatted as a DER-encoded ASN.1 data structure of the type `SubjectPublicKeyInfo` defined in the Internet X.509 certificate profile [HFPS02]. The auxiliary parameters are the following three unsigned integers:

- a 128-bit modifier,
- the 64-bit subnet prefix of the address, and
- an 8-bit collision count, which can have values 0, 1 and 2.

Two 128-bit hash values Hash1 and Hash2 are computed with the SHA-1 hash algorithm from these parameters. The input to Hash1 is the concatenation of the modifier, subnet prefix, collision count, and the encoded `SubjectPublicKeyInfo` structure. The 64-bit Hash1 is obtained by taking the leftmost 64 bits of the 160-bit SHA-1 hash value. The input to Hash2 is the same except that the subnet prefix and collision count are replaced with 64+8 zero bits. The 112-bit Hash2 is obtained by taking the leftmost 112 bits of the 160-bit SHA-1 hash value.

The crucial part in the way the hash inputs are defined is that the address owner's public key and the modifier are input to both hashes but the subnet prefix and collision count only go into Hash1. This means that the brute force search for 16*Sec zero bits in Hash2 can be done by varying the modifier, and without knowing the subnet prefix. Thus, the address owner does not need to redo the $O(2^{16*Sec})$ brute-force search when the subnet prefix changes. Also, if an address collision occurs, the address owner does not need to repeat the brute-force search in order to get another address.

5. CGA Algorithms and Protocols

This section explains how to generate, verify and use CGA addresses. An algorithm for address generation is given in Section 5.1 and an algorithm for verification in Section 5.2. Section 5.3 explains how the CGA addresses should be used for authenticating messages in security protocols.

5.1 Address Generation

The process of generating a new CGA takes three inputs: the 64-bit subnet prefix, the public key of the address owner as a DER-encoded ASN.1 structure of the type `SubjectPublicKeyInfo`, and the security parameter Sec, which is an unsigned 3-bit integer. The result of address generation is a new CGA address and the associated auxiliary parameters (as defined in Sections 4.1-4.2).

For any Sec value from 0 to 7, a CGA address can be created as follows:

1. Set the modifier to a random 128-bit value.

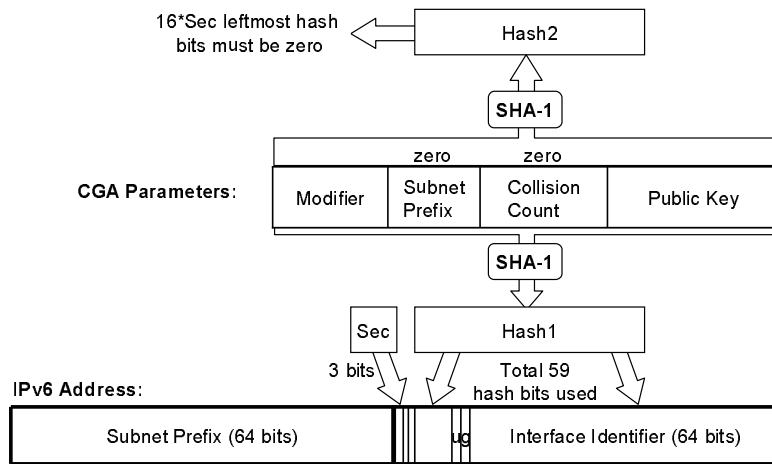


Fig. 2. Data flows in address generation

2. Concatenate the modifier, 64+8 zero bits, and the encoded public key. Execute the SHA-1 algorithm on the concatenation. The leftmost 112 bits of the result are Hash2.
3. Compare the 16*Sec leftmost bits of Hash2 with zero. If they are all zero (or if Sec=0), continue with the step 4. Otherwise, increment the modifier and go back to step 2.
4. Set the collision count value to zero.
5. Concatenate the modifier, subnet prefix, collision count and encoded public key values. Execute the SHA-1 algorithm on the concatenation. The leftmost 64 bits of the result are Hash1.
6. Form an interface identifier by setting the “u” and “g” bits in Hash1 both to 1 and the three leftmost bits to the value Sec.
7. Concatenate the subnet prefix and interface identifier to form a 128-bit IPv6 address.
8. If an address collision is detected, increment the collision count and go back to step 4. However, after three collisions, stop and report the error.

The output from the algorithm is the address and the auxiliary parameters (modifier, subnet prefix and collision count).

Choosing the initial value of the modifier randomly in step 1 reduces the probability of the same values being tried repeatedly in the brute-force search and enhances privacy by making multiple addresses generated from the same public key unlinkable to each other. The quality of the random number generator does not affect the strength of the CGA-based authentication.

The cost of generating a new address depends on the security parameter Sec. If one does not want to use the hash extension and sets Sec=0, no brute-force search is done. In that case, Hash2 need not be computed, and the security of the address depends on the attacker's inability match a 59-bit hash value. This is reasonable for client hosts that do not expect to be targets of expensive denial-of-service attacks, especially if

their address changes frequently and is not registered in DNS or other directory services. However, we feel that implementations should support the stronger addresses because they may be costly to update if the circumstances change.

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 2-3 takes, on the average, $O(2^{16*Sec})$ iterations to complete. (More accurately, the median number of iterations is $0.69*2^{16*Sec}$).

If the subnet prefix of the address changes but the address owner's public key does not, the old value of the modifier can be used and it is unnecessary to repeat the brute-force search. This is because the subnet prefix is not included in the input of Hash2.

The data flows in address generation are depicted in Figure 2. The address generator starts with the public key and the security parameter Sec. It varies the modifier until Hash2 contains the desired number of zeros. When computing Hash2, the subnet prefix and collision count are replaced with zero bits. Once a match for $16*Sec$ zeros in Hash2 is found, the generator computes Hash1 and the address.

5.2 Verification of Address Ownership

CGA verification takes as input an IPv6 address, a public key, and the auxiliary parameters (modifier, subnet prefix and collision count). The verification either succeeds or fails. If the verification succeeds, the verifier knows that the public key belongs to 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. Check that the collision count value is 0, 1 or 2, and that the subnet prefix value is equal to the subnet prefix (i.e. leftmost 64 bits) of the address. The CGA verification fails if either check fails.
2. Concatenate the modifier, subnet prefix, collision count and the public key. Execute the SHA-1 algorithm on the concatenation. The 64 leftmost bits of the result are Hash1.
3. Compare Hash1 with the interface identifier (i.e. the rightmost 64 bits) of the address. Differences in the “u” and “g” bits and in the three leftmost bits are ignored. If the 64-bit values differ (other than in the five ignored bits), the CGA verification fails.
4. Read the security parameter Sec from the three leftmost bits of the interface identifier of the address. (Sec is an unsigned 3-bit integer.)
5. Concatenate the modifier, 64+8 zero bits and the public key. Execute the SHA-1 algorithm on the concatenation. The leftmost 112 bits of the result are Hash2.
6. Compare the $16*Sec$ leftmost bits of Hash2 with zero. If any one of these is non-zero, CGA verification fails. Otherwise, the verification succeeds. If Sec=0, verification never fails at this step.

While using a Sec value above 0 is voluntary to the address owner, the address verifier must support all Sec values. First, the verification procedure requires a constant amount of computation and is relatively fast regardless of the value of Sec.

Second, if all verifiers did not support higher Sec values, there would hardly be any reason for an address owner to use them.

5.3 Using CGA Addresses in Security Protocols

In order to authenticate a message as coming from a specific CGA address, the address owner will sign the message with its public key. It will then send

- the message,
- the public key,
- the auxiliary parameters, and
- the signature.

from the CGA address to the verifier. The verifier will follow the steps outlined in the previous section to verify the address. If the address verification succeeds, the verifier knows that the public key belongs to the address owner. Finally, the verifier uses the public key to verify the signature on the message. If the signature is valid, the verifier knows that the message was sent by the owner of the specific IPv6 address.

CGA authentication may also be used as a part of a key-exchange protocol. The procedure is that same as above, except that the signed message belongs to a key-exchange protocol.

We have proposed reserving the currently unused bit combination $u=1, g=1$ for CGA addresses. The type bits would tell the recipient of an IP packet that the source address is a CGA address and that the sender supports CGA-based authentication. This is would make it possible to use authenticated CGA addresses and unauthenticated legacy address side by side. As long as no such type bits are allocated, CGA addresses will have to use the bit combination $u=0, g=0$ and it will be impossible to tell apart CGA addresses and non-CGA addresses. This makes it more difficult to deploy new security protocols based on CGA. In some situations, it is possible to accept both authenticated and unauthenticated information and to give priority to the authenticated over the unauthenticated.

6. CGA Properties and Limitations

This section explains the design choices made in specifying the CGA format. Security goals and techniques, including the hash extension, are discussed in Sections 6.1-6.2. Sections 6.3-6.4 are about other practical design considerations: mobility support and address collisions. Section 6.5 explains the privacy implications of using CGA addresses.

6.1 Security Goals and Limitations

The purpose of CGA addresses is to prevent stealing and spoofing of existing IPv6 addresses. The public key of the address owner is bound cryptographically to the

address. (The “public key” obviously includes a public exponent and all associated parameters that would normally be associated with the key, such as a modulus or a finite field.) The address owner can use the corresponding private key to assert its ownership of the address and to sign messages sent from the address. This is because the attacker would have to find a collision of the cryptographic hash value Hash1 in order to bind another private key to the address. (The property of the hash function needed here is called second pre-image resistance.)

It is important to understand that CGA-based authentication does not prove that a node with the authenticated address exists or that it has any kind of permission to use the specific subnet prefix. An attacker can create a new address from an arbitrary subnet prefix and its own public key and then sign messages from this new address. (In a way, the attacker is the owner of the new address.) What the attacker cannot do is to sign messages from somebody else's address.

The attacker can also create an address from someone else's public key. While the attacker cannot sign messages for that address, it could replay old messages signed by the key owner. Thus, a CGA-based signature does not necessarily imply that the key owner wants to use the specific address. Protocols that use CGA-based signatures will typically correct this problem because the recipient can see from the contents of the signed message that the key owner wants to be associated with a CGA address.

Another fundamental limitation of CGA-based authentication is 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 of the sender. That is, CGA-based authentication prevents spoofing of source IP addresses but it does not tell which address is the correct one. While CGA is not a complete authentication solution for all purposes, it does block one of the main avenues of attack against Internet protocols.

In many network-layer signaling protocols, such as neighbor discovery and Mobile IPv6, the IPv6 address is the natural identifier for a node. CGA-based authentication is obviously best suited for this type of protocols. If authentication of a higher-level identifier is desired, care must be taken to map the identifier securely to an IPv6 address.

6.2 Preventing Brute-Force Attacks

As computers become faster, the 64 bits of the interface identifier will not be sufficient to prevent attackers from searching for hash collisions. 64 bits (or 59 bits in our case) is sufficient to discourage large-scale attacks against multiple addresses with today's technology. It is not sufficient to guarantee security of protocols that should have a lifetime of tens of years.

The most worrying attack is one where the attacker pre-computes a large database of interface identifiers from its own public key(s) and uses this database to find matches for many addresses.

It helps somewhat that we include the subnet prefix of the address in the hash input. This prevents the attacker from using a single pre-computed database to attack addresses with different subnet prefixes. It needs to create a separate database for each subnet prefix. Link-local addresses are, however, left vulnerable because the same subnet prefix is used by all IPv6 nodes. Even global-scope addresses will

eventually become vulnerable if the exponential growth of attacker computing power and memory continues.

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 paper 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.

The hash extension is implemented with the second hash value Hash2. During address generation, the input to a second hash function Hash2 is modified by varying the value of modifier until the leftmost $16 \cdot \text{Sec}$ bits of Hash2 are zero. This increases the cost of address generation approximately by a factor of $2^{16 \cdot \text{Sec}}$. It also increases the cost of brute-force attacks by the same factor. That is, the cost of finding a modifier that binds the attacker's public key with somebody else's address is increased approximately by a factor of $2^{16 \cdot \text{Sec}}$ (i.e. from $O(2^{59})$ to $O(2^{59+16 \cdot \text{Sec}})$). The ratio between the cost of a brute-force attack and the cost of address generation remains at the constant value 2^{59} .

The effectiveness of the hash extension depends on the assumption that the attacker's and the defender's computational capacities will grow at the same (exponential) rate. This is clearly not the case if the defender generates the addresses on a small mobile device, such as a handheld organizer, where long battery life is far more important than raw computing power. However, there is no reason why the brute-force stage of the address generation should be done on the mobile device itself. Instead, steps 1-3 of the address generation algorithm in Section 5.1 should be delegated to a powerful server machine. After the results are transferred to the mobile device, it can continue from step 4 of the algorithm. The mobile device does not need to contact the server ever again, as it can generate new addresses without repeating the brute-force search whenever its address prefix changes or an address collision is detected (see Sections 6.3-6.4). The public-private key pair may be generated either on the mobile device or at the server.

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. When CGA addresses are used for denial-of-service protection or opportunistic key exchange, the practical approach is to use a small Sec value now and to increment Sec when denial-of-service attacks based on brute-force reversal of the hash become common.

6.3 Mobility Support

Mobile IPv6 nodes may move to insecure networks where they are at the risk of denial-of-service attacks, for example, during the duplicate address detection procedure (Section 7.1). Mobile IPv6 mobiles also need to send authenticated signaling messages from a specific IPv6 address (Section 7.2). Therefore, the CGA-based authentication and hash extension are particularly important for mobile nodes. The problem is that a mobile node needs to change its subnet prefix and, thus,

generate a new IP address when moving from network to network. If mobiles had to repeat the $O(2^{16 \cdot \text{Sec}})$ brute-force search every time, it would be impossible to use the hash extension (i.e. Sec values greater than 0).

For the above reason, we have specified CGA addresses in such a way that the subnet prefix can be changed without repeating the expensive part of address generation. We have not included the subnet prefix in the input of Hash2.

The result is weaker than if the subnet 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 without including the subnet prefix in either hash. It is also at least as strong as not using the hash extension but including the subnet prefix.

Theoretically, the above means that if $\text{Sec}=0$ and 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 globally routable addresses. This effect can be countered by setting $\text{Sec}=\log_2(N)/16$ for link-local addresses. If Sec is 1 or higher for all addresses, the difference between link-local and globally routable addresses becomes insignificant because the hash extension dominates the cost of attacks.

6.4 Address Collisions

The parameter collision count is used to modify the input to Hash1 if there is an address collision. Like the subnet prefix, the collision count can be modified during address generation without repeating the brute-force search.

It is important not to allow collision counts higher than 2. First, it is extremely unlikely that three collisions would occur. The reason is certain to be either a configuration or implementation error. (Or a denial-of-service attack if CGA-based proof of address ownership is not used. See Section 7.1.) 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 the modifier. Thus, the more different values are allowed for collision count, the less effective the hash-extension technique is in preventing brute-force attacks.

6.5 Address Revocation

If the private key of the address owner is compromised, the only remedy is to replace the key and the address. If the address owner uses CGA-based authentication for preventing denial-of-service attacks, it will be vulnerable to these attacks until the key has been replaced. On the other hand, if the authentication is used to protect secrets or to authorize service access, the attacker will be able to gain access to the secrets and services until the relevant correspondents have been notified of the address change.

Typically, however, the key compromise is a result of the device that uses the CGA address falling into wrong hands. After the device is lost, further denial-of-service attacks against its IPv6 address become irrelevant. If the same key is used for application level security, the situation is not different from the compromise of any device storing a secret authentication key. Well-known techniques, such as

passwords, PIN codes and smart cards can be applied to reduce the risk of such compromise.

6.6 Privacy Issues

CGA-based authentication has some implications for 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 public key to its correspondents, it should also replace the key at the same time as changing the address. This gives the same level of protection as the IPv6 address privacy extensions [ND01].

While the cost of generating new public keys and the cost of CGA address generation may imply less frequent address changes, this problem is mitigated by the fact that the expensive part of the address generation may be done in advance or offline. It should also be noted that the nodes that benefit most from high Sec values (e.g. DNS servers, routers, and data servers) usually do not require pseudonymity, while the nodes that have high privacy requirements (e.g. client PCs and mobile hosts) are unlikely targets for expensive brute-force attacks and can do with lower Sec values.

7. Applications

This section describes some CGA applications on which there is work in progress.

7.1 Secure Neighbor Discovery

The IPv6 address autoconfiguration [TN98], duplicate address detection (DAD), neighbor discovery (ND) [NN98], are threatened by denial-of-service attacks. The problem is that it is not clear who owns an IP address and who is authorized to control the mapping between an IP address and link-layer (MAC) addresses.

In stateless autoconfiguration, an IPv6 node picks an arbitrary IPv6 address in the network where it is located and sends a multicast message to check that nobody else is using the same address. An attacker can prevent other nodes from obtaining an IP address by responding to all duplicate address detection messages and claiming to be using any address that the target nodes happen to pick.

Neighbor discovery is the IPv6 equivalent to of ARP, i.e., a protocol for mapping IP addresses into link-layer addresses. A well-known problem is that an attacker can redirect packets away from their right next-hop destination by spoofing neighbor discovery messages.

CGA addresses can prevent the attacks described above. CGA addresses are particularly suitable for securing these protocols because the goal is to authenticate an IP address and not some higher-level identifier. A node with a CGA address can prove its ownership of the address by signing the DAD and ND messages.

7.2 Mobile IPv6

In the Mobile IPv6 [JPA03] draft specification, the mobile node informs its correspondents about its current location by sending binding updates (BU), which must be authenticated to prevent denial-of-service attacks. The current draft uses a relatively weak, non-cryptographic authentication method based on the assumption that some network routes are likely to be secure. An alternative would be to use CGA-based public-key authentication. CGA addresses would be particularly suitable for this purpose because Mobile IP uses an IP address as the only node identifier. CGA-based authentication could also reduce the number of protocol messages. Thus, there may be a case for specifying CGA-based authentication as an optional optimization.

7.3 Opportunistic IPsec

If both end nodes of a connection have a CGA address, they can use the CGA-based authentication in a key exchange and create an IPsec security association for encryption and data authentication. They can do this kind of authentication "opportunistically", i.e., whenever both end nodes support CGA. The benefits of the opportunistic protection include prevention of address and connection hijacking attacks, privacy against passive eavesdropping, and protection against filtering of IP packets by network intermediaries. A gateway-based system for opportunistic encryption is described in [CM02]. There is no reason why the same mechanisms could not be used host to host.

It is important to note that the two nodes are authenticating each other's IPv6 addresses, not the host names or users. Therefore, the CGA-based authentication prevents only IP source-address spoofing but not DNS spoofing where the attacker interferes with the mapping of domain names to IP addresses.

In the future, Secure DNS [Eas99] may give a reasonable assurance of the authenticity of IP addresses. Together, Secure DNS and CGA provide strong authentication of hosts by their domain names. An advantage of combining CGAs with Secure DNS is that the name service need not store the public keys. It only needs to provide a reliable binding between host names and addresses, which is its original function.

In Section 6.2 we argued that when CGA-based authentication is used to prevent denial-of-service attacks, the protection does not need to be absolute. The same applies to opportunistic authentication and encryption. That is, the protection only needs to be strong enough to make attacks uneconomical. Thus, it suffices to use a low Sec value and to increment it when attacks become common. If CGA is used as a component of a strong general-purpose authentication, the situation is quite different. In that case, a wider security margin is needed and higher Sec values become relevant much sooner.

8. Conclusion

We described a new format for cryptographically generated addresses (CGA). They are IPv6 addresses where the interface identifier contains a hash of the address owner's public key. The format for the addresses and associated parameters are generic so that they can be easily used in security protocols where authentication of an IPv6 address is desired. We gave detailed algorithms for address generation and verification. We extended the effective hash length beyond the 64-bit limit of earlier proposals and, thus, removed a major security weakness that has previously prevented widespread adoption of the CGA addresses. The hash extension technique enables the use of CGA-based authentication in protocols, such as neighbor discovery, which must remain secure for tens of years. We also described promising applications for CGA-based authentication.

Acknowledgments

Many of the ideas in this paper were influenced by Michael Roe, Christian Huitema, Pekka Nikander and participants in the IETF Mobile IP and Secure neighbor Discovery Working Groups.

References

- [Aur02] Tuomas Aura. Mobile IPv6 security. In Proc. Security Protocols, 10th International Workshop, Cambridge, UK, April 2002. To appear in Springer LNCS.
- [BFIK99a] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos Keromytis. The KeyNote trust-management system version 2. RFC 2704, IETF Network Working Group, September 1999.
- [CM02] Claude Castelluccia and Gabriel Montenegro. IPv6 opportunistic encryption. Technical Report 4568, INRIA, October 2002.
- [Eas99] Donald Eastlake. Domain name system security extensions. RFC 2535, IETF Network Working Group, March 1999.
- [EFL+99] Carl Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. SPKI certificate theory. RFC 2693, IETF Network Working Group, September 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.
- [JPA03] David B. Johnson, Charles Perkins, and Jari Arkko. Mobility support in IPv6. Internet-Draft draft-ietf-mobileip-ipv6-24.txt, IETF Mobile IP Working Group, June 2003. Work in progress.
- [KS99] Phil Karn and William A. Simpson. Photuris: session-key management protocol. RFC 2522, IETF Network Working Group, March 1999.

- [MKKW99] David Mazierès, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating key management from file system security. *Operating Systems Review*, 34(5):124-139, December 1999.
- [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 2002.
- [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.
- [NYW03] Pekka Nikander, Jukka Ylitalo, and Jorma Wall. Integrating security, mobility, and multi-homing in a HIP way. In *Proc. Network and Distributed Systems Security Symposium (NDSS'03)*, pages 87-99, San Diego, CA USA, February 2003.
- [ODG+02] Satomi Okazaki, Anand Desai, Craig Gentry, James Kempf, Alice Silverberg, and Yiqun Lisa Yin. Securing MIPv6 binding updates using address based keys (ABKs). Internet-Draft draft-okazaki-mobileip-abk-01.txt, October 2002. Work in progress.
- [OR01] Greg O'Shea and Michael Roe. Child-proof authentication for MIPv6 (CAM). *ACM Computer Communications Review*, 31(2), April 2001.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proc. CRYPTO 84*, volume 196 of LNCS, pages 47-53, Santa Barbara, CA USA, August 1984. Springer-Verlag.
- [TN98] Susan Thomson and Thomas Narten. IPv6 stateless address autoconfiguration. RFC 2462, IETF Network Working Group, December 1998.