
Wireless Personal Communications **29:** 221–232, 2004.

Cryptographically Generated Addresses for Constrained Devices*

CLAUDE CASTELLUCCIA

INRIA Rhône-Alpes 655, avenue de l'Europe 38330 Montbonnot, France
E-mail: claude.castelluccia@inria.fr

Abstract. Cryptographically Generated Addresses (CGAs) have been designed to solve the so-called IPv6 *Address Ownership* problem. The current IETF CGA proposal relies on RSA signature. Generating an RSA signature is quite expensive and might be prohibitive for small devices with limited capacities. For example, a 1024-RSA signature requires approximately 1536 modular multiplications.

In this paper, we propose a new CGA scheme whose verification requires fewer than 10 modular multiplications. We achieve this performance gain by (1) selecting an efficient signature scheme, namely the small prime variation of the Feige-Fiat-Shamir scheme and (2) tuning the cryptographic parameters of this signature scheme to the security strength of the CGA (i.e. the size of the hash function used to generate it).

Keywords: CGA, IPv6 security, mobile IPv6, address ownership

1. Introduction

A Cryptographically Generated Address (CGA) [1, 2] is an IPv6 address whose Interface Identifier (the 64 lower bits) is generated by hashing the address owner's public key. A host proves ownership of a CGA address by proving that it knows the private key associated with it. This is typically achieved by signing a challenge.

CGAs have generated a lot of interests lately at the IETF (Internet Engineering Task Force) and is currently being standardized [1, 3]. Several working groups, such as Mobile IP[1] or SEND[2], are considering to use CGAs to solve various IPv6 security problems. CGAs have also shown to be very useful in mobile ad-hoc environments because they do not rely on any public key infrastructure or third trusted party [4, 5]. They can also be used to bootstrap a security association between two nodes as shown in [1, 6, 7].

However, in wireless environments, nodes tends to have limited capacities (CPU, battery) and the signature cost (generation and verification) to prove or verify a CGA can sometime be overwhelming. In particular, RSA-based CGAs are very demanding for the provers because signing with RSA is very costly. A more efficient scheme is required.

In this paper, we propose a new CGA scheme that uses a variation of the Feige-Fiat-Shamir signature [8]. We show that with this scheme a CGA proof of ownership requires fewer than 10 modular multiplications (when precomputation is used) for the prover and two modular multiplications (and a hash computation) for the verifier. For the same level of security, an RSA-based CGA scheme respectively requires 1536 and two modular multiplications. We achieved

*Work performed while the author was visiting UC, Irvine, USA.

[1] IP Routing for Wireless/Mobile Hosts WG, http://www.ietf.org/html.charters/mobileip-charter.html.

[2] Securing Neighbor Discovery WG, http://www.ietf.org/html.charters/send-charter.html.

this performance gain by: (1) selecting an efficient signature scheme, namely the small prime variation of the Feige-Fiat-Shamir scheme and (2) tuning the cryptographic parameters of this signature scheme to the security strength of the CGA address (i.e. the size of the hash function used to generate it).

2. IPv6 Cryptographically Generated Addresses

2.1. OVERVIEW

CGAs have been proposed to solve the IPv6 *address ownership* problem [9], and are generally useful to secure address auto-configuration and redirect operations in numerous protocols [1–3, 10]. For example, in Mobile IPv6 [11], a node starts using its home address, and, each time it moves to a different link, it issues a Binding Update that specifies its current address. The issue is in handling these Binding Updates securely. Why should the correspondent node believe the mobile node when it claims that it does, in fact, own the home address contained in the binding update? The risk is that this mobile node could be issuing a redirect for another node's home address in order to redirect its packets. Ignoring this address ownership problem can lead to *denial-of-service* (DoS) and unauthorized redirect attacks.

The current Mobile IPv6 specification uses a procedure called *Return Routability Test* to authorize the establishment of the binding between a home address and a care-of address. This procedure enables the correspondent node to verify that the mobile node is really addressable at its claimed care-of address as well as at its home address. This is done by testing whether packets addressed to the two claimed addresses are routed to the mobile node. The mobile node can pass the test only when it is able to supply proof that it received certain data which the correspondent node sends to those addresses. However, this solution is known to be weak since it does not protect against an attacker who can eavesdrop packets sent between the mobile node and its home agent and packets sent between the mobile node and its correspondent node. In contrast, CGAs provide a much stronger solution.

CGAs are also very useful in securing IPv6 neighbor and router discovery procedures [10]. For example, in IPv6, after a node auto-configures an IPv6 address it needs to run a duplicate address detection (DAD) protocol to verify that its address is not being used by another node on the network. This protocol is susceptible to Denial of Service attacks because any malicious node can pretend to own this address. Similarly, a node that changes its IPv6 address can send a redirection message to the routers. But, why should the routers believe that this message was originated by the owner of the address being redirected? With a CGA, a node can actually prove that it owns its address. This proof can be used to resolve conflicts when two nodes claim to own the same address and to avoid redirection attacks.

A CGA scheme associates a public-private key pair, referred in this paper as PK/SK, to each host and derive its IPv6 addresses from it [1, 2]. A CGA is a valid IPv6 address. The top 64 bits are the host's routing prefix as in [12]. The bottom 64 bits, the *Cryptographically Generated Host Identifier* (CGHID), are derived from the host's public key as follows:

$$\text{CGHID} = hash(\text{PK}||network\_prefix) \tag{1}$$

Where $hash(\cdot)$ is a hash function, such as SHA1 [20], *network_prefix* is the host's network prefix and PK is the public key associated with the host.

These addresses and identifiers have two very important properties:

1. They are *statistically unique*, because of the collision-resistance property of the cryptographic hash function used to generate them.
2. They are *securely bound to a given node*: the node can prove ownership of its CGA by publishing its public key, PK, and signing a challenge with its private key, SK. Any other node can verify the ownership of the CGA by verifying that the address was generated from PK and that the signature is valid. The prover's public key does not need to be certified. As a result, CGAs do not require any centralized security service such as a public key infrastructure or a key distribution center.

These characteristics (1) make CGAs a very scalable naming system, and (2) provide an auto-configurable and solid foundation for nodes to engage in verifiable exchanges with each other.

2.2. SECURITY

There exist two possible ways for an attacker to steal an existing CGA address:

1. The attacker can retrieve the private key associated with the public key used to generate the target CGA. This attack is very difficult to perform if the underlying signature scheme is secure.
2. The attacker can find a public/private key pair whose public key hashes to the target CGHID. This attack is probably easier since the size of the CGHID is limited to 59 bits. The security of a CGA can actually be increased by using two hash functions [3]. The first one is used to create the CGHID as described above. The second one, of size $12 \times sec$ where sec is an integer, is used to increase artificially the cost of brute-force attacks. This hash must result in $12 \times sec$ zeros. This proposal adds some burden to the address configuration process because a host has to find a public key that hashes, with the second function, to $12 \times sec$ zeros. However, it increases the CGA security considerably because, in addition to matching the CGHID with the first hash function, an attacker must now match the $12 \times sec$ zero bits with the second hash.

3. Proposal Description

The current IETF CGA solution is based on the RSA signature scheme [3]. However, as shown in Section 5.1, the cost of generating a signature is high and might be prohibitive for constrained devices.

We propose a new CGA scheme that uses the small prime variation of the Feige-Fiat-Shamir signature scheme [8, 13, 14]. As we will see, the use of this scheme has several benefits: (1) It requires less processing power from the signer and verifier. (2) Its security (and therefore cost) can be adjusted to the size of the hash used to generate the CGA Interface ID. Tuning the size leads to a better security/performance trade-off.

The rest of this section reviews the Feige-Fiat-Shamir (FFS) signature scheme, its small prime variation (MFFS) and then describes our new CGA proposal.

3.1. The Feige-Fiat-Shamir (FFS) Signature Scheme

As other signature schemes, the FFS has three steps: (1) the key generation, (2) the signature generation and (3) the signature verification steps.

Each of these three steps is described below:

1. *Key Generation*. To generate its public and private keys, entity A should do the following:
   (a) Generate random distinct secret primes p, q and form $n = p \times q$.
   (b) Select a positive integer k and distinct random integers $s_1, s_2, \ldots, s_k \in \mathbb{Z}_n^*$.
   (c) Compute $v_j = s_j^{-2} mod n$, $1 \leq j \leq k$.
   (d) A's public key is the k-tuple $(v_1, v_2, \ldots, v_k)$ and the modulus n; A's private key is the k-tuple $(s_1, s_2, \ldots, s_k)$.
2. *Signature Generation*. To sign a message, m, entity A should do the following:
   (a) Select a random integer r, $1 \leq r \leq n-1$.
   (b) Compute $u = r^2 mod n$.
   (c) Compute $e = (e_1, e_2, \ldots, e_k) = h(m||u)$; each $e_i \in \{0, 1\}$.
   (d) Compute $s = r \cdot \prod_{j=1}^{k} s_j^{e_j} mod n$.
   (c) A's signature for m is (e, s).
3. *Signature Verification*. To verify A's signature (e, s) on m, B should do the following:
   (a) Obtain A's authentic public key $(v_1, v_2, \ldots, v_k)$ and the modulus n.
   (b) Compute $w = s^2 \cdot \prod_{j=1}^{k} v_j^{e_j} mod n$.
   (c) Compute $e' = h(m \,||\, w)$.
   (d) Accept the signature if and only if $e = e'$.

The FFS scheme has been proved secure in the random oracle model. In fact [15] proves that if an existential forgery of the FFS signature is possible with non negligible probability, then factorization of the RSA moduli can be performed in polynomial time.

3.2. The Small Prime Variation of the Feige-Fiat-Shamir Signatures

Micali and Shamir proposed a variation of the Feige-Fiat-Shamir signature scheme that aims to reduce the signature verification time and the size of the public key [8]. We refer to this scheme as the MFFS scheme in the rest of this paper. This variation only affects the key generation phase. The signature generation and verification phases are the same as in Section 3.1.

In the new scheme, to generate its public and private keys, entity A should do the following:

1. Generate random distinct secret primes p, q and form $n = p \times q$.
2. Select a positive integer k and a set of k distinct *small* primes $v_1, v_2, \ldots, v_k \in Q_n$ (i.e. all the v_i have a square root in $\mathbb{Z}_n^*$).
3. Selects one of the four square roots s_j of v_j^{-1} mod n for each j, $1 \leq j \leq k$, i.e.: $s_j = (v_j)^{-1/2} \bmod n$.
4. The $s_1, s_2, \ldots, s_k$ form A's private key. The public key consists of n and the value v_1, $v_2, \ldots, v_k$.

The signature generation performance remains unchanged but signature verification becomes much more efficient since multiplications only involve small factors. Moreover, for the same reason, this variation reduces the public key size significantly.

3.3. THE MFFS-CGA SCHEME

In our proposal, a host A computes its CGA and related materials as follows:

1. A computes two large prime numbers p and q and computes $n = p \times q$.
2. It generates its CGA address as follows:
   CGHID = $hash(n \mid net\_prefix)$, where *net_prefix* is its home network prefix.
   Note that this definition is the same as the standard CGA scheme (described in Section 2) if n is replaced by PK. An RSA public key is defined by the tuple (e, n) where e is the public exponent and n is the modulus. However it is often recommended, for performance reasons, to use $e = 3$. Therefore, even in the RSA-CGA scheme, the CGHID can be defined as $hash(n \mid prefix)$, without losing any security. As a result, the same CGA address can be used for the two schemes. This is important for compatibility. For example if, for a given CGA, a host (the prover) supports both RSA and MFFS schemes but the verifier only supports the RSA-CGA scheme, then the verifier and prover can agree on using RSA. However, if the verifier does also support MFFS, the prover and verifier can agree on using the MFFS-CGA scheme. Compatibility between the RSA-CGA and MFFS-CGA schemes can therefore be achieved.
   A CGA scheme based on the standard Feige-Fiat-Shamir signature using the above CGA definition would be insecure. In fact, as described in Section 3.1, the factorization of n is not needed in order to compute a valid private and public key pair. As a result, a malicious party could trivially pretend to own any CGA address since it can generate a valid public/private key pair for it. With the standard Feige-Fiat-Shamir signature scheme, a CGA address must also use the public key v_i in the address generation phase.
3. It selects a positive integer k and the k first small prime numbers, v_i, that belongs to Q_n (i.e. v_i are quadratic residues in $\mathbb{Z}_n^*$ or in other words have square roots in $\mathbb{Z}_n^*$).
   Note that since n is the product of two prime numbers, p and q, the number of quadratic residues in $\mathbb{Z}_n^*$ is equal to $1/4 \times (p - 1) \times (q - 1)$ which is approximatively equal to $1/4 \times n$. Therefore, on average, one out of every four prime numbers is a quadratic residues in $\mathbb{Z}_n^*$.
4. It computes its private key s_i from the v_i as explained in Section 3.2.

When a node needs to prove the ownership of its address, it sends a message that contains its public key and its modulus n and signs this message with it private key using the MFFS scheme as in the standard CGA scheme [3]. When the verifier receives the packet, it verifies that (1) the CGA was generated from the prover's public key (included in the message) and (2) that the signature is correct. This protocol can easily be extended with a Diffie-Hellman exchange if necessary (see [1] for more details about the protocol).

4. Security Analysis

CGA addresses prevents from *address-spoofing* (and impersonation). It is important to realize that nothing prevents a malicious host from generating a fake CGA address and using it. However it is very difficult for a malicious to pretend to own someone-else's address (i.e. to steal an address).

There are essentially three ways a malicious host can steal a MFFS-CGA address and impersonate its victim:

1. By factorizing n: A malicious node that can factorize n can compute a valid public/private key pair for the modulus n and therefore steal the CGA address derived from it. However if the size of n is large (i.e. 1024 bits), finding the factorization of n is considered to be very difficult.
2. By computing the square root of the public key: A node that can compute the square root of the public key can retrieve the corresponding private key. In the MFFS scheme, the private key is composed of the *first k* prime numbers that are in Q_n. Computing the square root of a random number modulus a large composite number, without knowing its factorization, is considered to be a difficult problem [16]. Although it has not been proved, the computation of the square root of a *small* prime number modulus a large composite number is also believed to be difficult [8].
3. By finding a public/private key pair whose public key (n', v') hashes to the victim's CGA: If the CGHID length is large enough (as explained in Section 2.2) this should not be a problem. Note that increasing the CGHID length only affects the performance of the MFFS-CGA Generation phase. The performance of the MFFS-CGA proof of ownership generation and verification (which are performed on-line and are therefore critical) are unaffected.
4. By forging a MFFS signature: A malicious node can impersonate a node (i.e. pretend to own the node's CGA) by forging a signature. The probability of forging a signature for an arbitrary message is $1/2^{k-1}$, i.e. an adversary would have to try, on average, 2^{k-1} random values r to find a forged signature. Therefore the larger k is, the more difficult it is to forge a MFFS signature. A natural tendency would be to choose a large number for k. However, it is useless and actually harmful to over-dimension k in a MFFS-CGA for at least two reasons:
   (a) The security of a CGA depends on the CGHID size i.e. the size of the hash used to generate the Interface Identifier. This size is $59 + 12 \times sec$ as shown in Section 2. Consequently it is useless to use a value of k larger than $59 + 12 \times sec$ since the attack on the CGA hash function, described above, would then be easier and would probably become the target of the attacker. The parameter k must therefore be set to $59 + 12 \times sec$.
   (b) As shown in Section 5.1, the performance of MFFS depends on the value of k. A large value for k increases the signature security but degrades significantly its performance.

5. Performance Evaluation

5.1. Processing Performance

To prove ownership of a CGA, the prover has to sign a message, such as a Mobile IP Binding Update. To verify ownership of a CGA, a verifier has to compute the hash of a public key and to verify a signature. As we will see later in this section, the cost of the hash is negligible compared to the cost of verifying or generating a signature. As a result, the cost of a CGA scheme depends directly on the performance of the underlying signature. In the rest of the section, we compare the performance of the RSA, DSA and MFFS signature schemes.

In the MFFS scheme, signing and verifying a message only require $(k/2 + 1)$ modular multiplications. Moreover, the signing performance can be improved with precomputation, as

Table 1. Modular multiplications

Algorithm	Sign	Verify
RSA $\|n\| = 1024$	1536[0] mod 1024	2 mod 1024
DSA $\|q\| = 1024$ $\|p\| = 160$	2[240] mod 160	480 mod 1024
MFFS $\|k\| = 59$	31[1] mod 1024	$1-2$ mod 1024
MFFS $\|k\| = 71$	37[1] mod 1024	$1-2$ mod 1024
MFFS $\|k\| = 59; y = 4$	8[1] mod 1024	$1-2$ mod 1024
MFFS $\|k\| = 71; y = 4$	9[1] mod 1024	$1-2$ mod 1024

suggested in [17]. In fact, to sign a message a prover computes $s = r.\prod_{j=1}^{k} s_j^{e_j} mod n$. Since the $s_i's$ do not change from message to message, and the $e_j's$ are either zero or one, the different possible products $\prod_{j=1}^{k} s_j^{e_j}$ can be precomputed and stored by the signer. For large k, it might not be practical to precompute all these values. Instead the s_i can be partitioned into smaller sets of y elements and precomputes each of them. For example, if each subset contains four elements (i.e. $y = 4$), the first subset will contain all the possible products of $s_1^{e_1} \times s_2^{e_2} \times s_3^{e_3} \times s_4^{e_4}$, the second subset all the possible products of $s_5^{e_5} \times s_6^{e_6} \times s_7^{e_7} \times s_8^{e_8}$ and so on. The number of modular multiplications to sign a message can then be reduced by k/y. The storage cost is then $k/y \times (2^y - 1)$ products, i.e. $k/y \times (2^y - 1) \times |n|$ bits. For example, if $|n|$ is 1024, k is 59 and y is 4, the signature generation time is reduced by four for an extra cost of 226.3 kbits (i.e. about 28 kBytes) of data memory.

Table 1 compares the number of modular multiplications required by RSA, DSA and MFFS (and its optimizations) to generate and verify a signature. These estimates, which were derived from [18], have been obtained from naive algorithms and can therefore be optimized. All the multiplications are 1024-bit modular multiplications except for DSA that uses 160-bit modular multiplications for signing. The values within brackets indicate the numbers of multiplications that can be precomputed. Verification with MFFS requires one modular multiplication and $k/2$ simple (i.e. non-modular) multiplications with small factors. The total cost is equivalent to 1 or 2 modular multiplications [18].

These results show that our MFFS-CGA scheme is much more efficient than an RSA or DSA-based scheme. A host can prove ownership of its CGA with fewer than 10 modular multiplications (with precomputation). Indeed, a MFFS signature generation requires, on average, $k/2$ modular multiplications. With precomputation, this cost can be reduced to k/y i.e. to fewer than 10 modular multiplications when $|k| = 71$ and $y = 4$. Address ownership verification only requires 2 modular multiplications.

To verify these results, we implemented the MFFS signature scheme under OpenSSL (Openssl,). We then measured the performance of the key generation, signature generation and signature verification phases for different values of k and size of n. We compared these results with the ones obtained with RSA and DSA (with precomputation).

Table 2. Performance comparison-1024 bits

Algorithm	KeyGen (s)	Sign (μs)	Verify(μs)
RSA	0.261	7279	298
DSA	1.7	57.5	3612
MFFS $\|k\| = 56$	1.73	1040 (828)	187
MFFS $\|k\| = 72$	2.16	1669 (845)	207
MFFS $\|k\| = 88$	2.6	1810 (950)	222
MFFS $\|k\| = 104$	3.0	1966 (1088)	242
MFFS $\|k\| = 120$	3.56	2297 (1292)	257

Table 3. Performance comparison-2048 bits

Algorithm	KeyGen (s)	Sign (μs)	Verify (μs)
RSA	1.88	39873	627
DSA	17	62.5	11926
MFFS $\|k\| = 56$	10.5	3559 (1889)	385
MFFS $\|k\| = 72$	13.0	4608 (2534)	406
MFFS $\|k\| = 88$	15.0	5433 (3408)	430
MFFS $\|k\| = 104$	18.0	6562 (3607)	448
MFFS $\|k\| = 120$	19.6	7262 (4031)	481

These measurements were performed on a 1.2 MHz Pentium III-M with 256 MB of RAM. Tables 2 and 3 display the results we obtained respectively for $|n| = 1024$ and $|n| = 2048$. For the MFFS measurements, values within parentheses in the *sign* column indicate the costs to sign a message when the precomputation optimization is used.

The results show that:

- DSA signature generation cost is very low but its verification cost is overwhelming. DSA is probably not a good candidate for a CGA scheme since it puts too much burden on the CGA verifier. This is not desirable since (1) in many CGA applications, such as Secure Neighbor Discovery [10], a given CGA proof can be verified by several verifiers and (2) this high cost could be used to perform some DoS attacks on the verifiers.
- RSA signature verification cost is very low but its signature generation cost is overwhelming. This is not desirable either since CGA's provers are often mobile nodes with limited capacities (energy, processing).
- MFFS scheme provides the best signature generation/verification cost trade-off. MFFS signature generation is more expensive than DSA but much cheaper than RSA. Its signature

Table 4. SHA1 performance

Len (bytes)	Time (μs)
100	6
300	9
500	12

verification cost is very low and even lower than RSA verification cost. If we compare the sum of the costs of signing and verifying a signature, i.e. *cost* (*sign*) + *cost* (*verification*), for a CGA size of 72 bits ($k = 72$) and a modulus of 1024 bits, this cost is equal to 1052 μs for MFFS (with precomputation), 7577 μs for RSA and 3669 μs for DSA. Therefore, a MFFS-CGA scheme is respectively 7 and 3.5 times more efficient than a RSA-CGA and a DSA-CGA scheme. For a modulus of 2048 bits, this cost increases to 40500 μs for RSA, 11988 μs for DSA and 2940 μs for MFFS. In this case, a MFFS-CGA scheme is respectively 18 and 6 times more efficient than an RSA-CGA and a DSA-CGA scheme.
- RSA key generation cost is much lower than DSA and MFFS costs. However, the keys can be precomputed off-line (for example when the node is charging its battery) or by a trusted device (for example, the host's home agent). Therefore, this cost is not critical.

Table 4 displays the cost of SHA1 for different input message sizes. These results confirm our initial assumption that the cost of the hashing in a CGA verification is negligible compared to the cost of the associated signature generation and verification.

5.2. Bandwidth Cost

In this section, we compare the bandwidth and storage costs of each signature scheme.

In the MFFS scheme, the public key is composed of the value n and of the k small prime numbers v_i. Since each prime will require around 2 bytes, the size of the public key is equal to $|n| + 16 \times k$ bits. The private key size is equal to $|n| \times |k|$ bits and the size of a signature is $(|n| + k)$ bits.

In comparison:

- RSA public key, private key size and the signature size is $|n|$.
- DSA public key size is $2 \times |p|$ (where $|p|$ is typically 1024), the private key size is $|q|$ (where $|q|$ is typically 160) and the signature size is $2 \times |q|$ (i.e. 320).

Table 5 provides a comparison of the public key (PK), private key (SK) and signature sizes (Sig.) of the RSA, DSA and MFFS signature schemes for $|n| = 1024$ and different values of k.

The DSA-based solution leads to shorter signatures than MFFS and RSA. MFFS requires a larger private key than RSA and DSA. However, since the private key is never sent, this has no effect on the bandwidth. The public key of DSA and MFFS are comparable and about the double of the RSA's public key size. However, as mentioned in [8], the public key size in the MFFS scheme can be reduced to 1024 bits using the *perturbation* method. With this method, n is defined as the product of two primes p and q such that p is congruent to 3 mod 8 and q is congruent to 7 mod 8. Then, it is guaranteed that for any $x \in \mathbb{Z}_n^*$ either x, $-x$, $2x$ or $-2x$ is a quadratic residue. As a result, the k small primes used in MFFS can just be the first k prime numbers. Since they are universal, they do not need to be part of the public key and, as a result,

Table 5. Bandwidth cost (bits)

Algorithm	PK	SK	Sig.	PK+Sig.	Stor.
RSA $\|n\| = 1024$	1024	1024	1024	2048	1024
DSA $\|q\| = 1024$ $\|p\| = 160$	2048	160	320	2368	160
MFFS $\|k\| = 59$	1968 (1024)	60416	1083 (1091)	3051 (2115)	60416
MFFS $\|k\| = 71$	2160 (1024)	72704	1095 (1103)	3255 (2127)	72704
MFFS $\|k\| = 59; y = 4$	1968 (1024)	60416	1083 (1091)	3051 (2115)	286976
MFFS $\|k\| = 71; y = 4$	2160 (1024)	72704	1095 (1103)	3255 (2127)	345349

the public key is only composed of the modulus n. If we note v_i (for $i = 1, \ldots, k$) the first k primes, then for any v_i either v_i, $-v_i$, $2.v_i$ or $-2.v_i$ is a quadratic residue. The corresponding private key component, s_i, can then be set to the inverse square root of this quadratic residue. Since the signer will either use v_i, $-v_i$, $2.v_i$ or $-2.v_i$ and the verifier will always use v_i, there will be an error between the signature computed by the signer and the one computed by the verifier. Indeed, the verifier will compute the value $w' = s^2 \cdot \prod_{j=1}^{k} v_j^{e_j} mod n$, whereas the correct value is $w = s^2 \cdot \prod_{j=1}^{k}((-1)^{b_j}.(-2)^{c_j}.v_j)^{e_j} mod n$, where b_j and c_j are either 0 or 1 according whether v_j, $-v_j$, $2.v_j$ or $-2.v_j$ was used to generate the signature. To solve this problem, the signer sends together with its signature the value the verifier must multiply w' to recover w. This value is equal to $\prod_{j=1}^{k}((-1)^{b_j}.(-2)^{c_j})^{e_j}$. This optimization reduces the public key size to 1024 but increases the signature size by $1 + \log_2(k)$ bits, i.e. by 8 bits when k is smaller than 128.

Note that in a CGA-based protocol, a node proves the ownership of its address by signing a challenge and including its public key in the reply. The verifier can then verify that the signature is fresh and that the IPv6 address was generated from the public key included in the reply. As a result, when comparing the performance bandwidth of different schemes it is important to compare the sum of the public key and the signature sizes. Table 5 compares these values for the different signature schemes (the values within the parentheses are the performance when the above perturbation optimization is used). The results show that the RSA-based solution has similar bandwidth performance than the optimized MFFS scheme. The DSA-based scheme requires a bit more bandwidth because of the large public key size.

Table 5 also displays the storage cost (stor.) of each signature scheme (this cost includes the secret key and the precomputation storage costs). These results show that an MFFS-based solution requires more storage memory than an RSA or DSA-based solution. For the most memory expensive solution (FFS with $|k| = 71$ and $y = 4$), 42 k Bytes of storage memory is required. Given that current PalmPilot comes with at least 2 MB of RAM[3] and considering the significant gain in performance, we believe that this cost is quite acceptable.

[3] http://www.palmos.com/dev/tech/hardware/compare.html.

6. Conclusions and Future Work

This paper proposes an efficient CGA protocol that is based on the small prime variation of the Feige-Fiat-Shamir signature scheme.

The security strength of a CGA can be tuned by varying the length of the hash function ($59 + 12 \times sec$ bits) used to generate the CGA from the Public Key [3]. With our proposal, the strength (and consequently the cost) of the associated signature is adjusted to the CGA strength. This leads to a very efficient CGA scheme. With the proposed scheme, a node can prove ownership of its address with fewer than 10 modular multiplications. In comparison, an RSA-based solution requires 1536 modular multiplications.

Although our scheme improves the performance of the RSA-based scheme it does not reduce the bandwidth overhead. A CGA scheme based on a signature scheme that uses small keys and signatures might be worthwhile studying from an energy prospective. Indeed it has been shown that the wireless transmission of one bit can require over 1000 times more energy than a single 32-bit computation [19]. Reducing the number of operations is therefore not sufficient, it is also necessary to reduce the number of transmitted bits. A CGA scheme that uses small key and signature might therefore result in a more energy-efficient solution. Our future work will consider the use of the elliptic curve digital sigdature algorithm (ECDSA) for CGA. Signature schemes based on elliptic curves are attractive for this application because they usually lead to much smaller key and signature sizes.

Acknowledgements

The author would like to thank Hahnsang Kim, Gabriel Montenegro, Pekka Nikander, Gene Tsudik, Shouhuai Xu and the anonymous reviewers for their helpful comments and suggestions about this work. The author is particularly grateful to one of the reviewers for suggesting the use of the perturbation optimization to reduce the public key size.

References

1. G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses", in *NDSS'02*, February 2002.
2. G. O'Shea and M. Roe, "Child-Proof Authentication for MIPv6 (CAM)", *ACM Computer Communications Review*, April 2001.
3. T. Aura, "Cryptographically Generated Addresses (CGA)", in *6th Information Security Conference (ISC'03)*, Bristol, UK, October 2003.
4. C. Castelluccia and G. Montenegro, "Protecting AODVng Against Impersonation Attacks", *ACM Mobile Computing and Communications Review*, July 2002a.
5. C. Castelluccia and G. Montenegro, "Dynamic and Secure Group Membership in Adhoc and Peer-to-Peer Networks", *ACM Mobile Computing and Communications Review*, July 2002b.
6. R. Bobba, L. Eschenauer, V. Gligor and W. Arbaugh, "Bootstrapping Security Associations for Routing in Mobile Ad-hoc Networks", May 2002.
7. S. Capkun, J.P. Hubaux and L. Buttyan, "Mobility Helps Security in Ad Hoc Networks", in *Proceedings of MobiHOC 2003*, Annapolis, p. 11, June 2003,
8. A. Micali and A. Shamir, "An Improvement on the Fiat-Shamir Identification and Signature Scheme", in *CRYPTO '88*, 1988, pp. 244–247.
9. P. Nikander, *An Address Ownership Problem in IPv6*, IETF, Draft-nikander-ipng-address-ownership-00.txt, February 2001.

10. J. Arkko, T. Aura, J. Kempf, V. Mntyl, P. Nikander and M. Roe, "Securing IPv6 Neighbor and Router Discovery", in *Wireless Security Workshop (WiSe2002)*, Atlanta, GA, September 2002.
11. D. Johnson, C. Perkins and J. Arkko, *Mobile IP for IPv6*, IETF, draft-ietf-mobileip-ipv6-24.txt, June 2003, (work in progress).
12. T. Narten and R. Draves, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, IETF, RFC3041, January 2001.
13. A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems", in *Advances in Cryptology: Proc. Crypto'86*, Springer, pp. 186–194, 1986.
14. U. Feige, A. Fiat and A. Shamir, "Zero Knowledge Proofs of Identity", *Journal of Cryptology*, 1988.
15. D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes", *Lecture Notes in Computer Science*, Vol. 1070, p. 387+, 1996.
16. A.J. Menezes, P.C. Van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
17. C. Wong and S. Lam, "Digital Signatures for Flows and Multicasts", *IEEE/ACM Transactions on Networking*, ACM Press, Vol. 7, 1999.
18. G. Poupard and J. Stern, "On the Fly Signatures Based on Factoring", in *ACM Conference on Computer and Communications Security*, pp. 37–45, 1999. "OpenSSL projet, http://www.openssl.org/,".
19. K. Barr and K. Asanovic, "Energy Aware Lossless Data Compression", in *Proceedings of MobiSys 2003*, San Francisco, May 2003.
20. NIST, http://www.itl.nist.gov/fipspubs/fip180-1.htm, NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.

Claude Castelluccia is a research scientist at INRIA, France. He is currently visiting the Secure Computing and Networking Center of the University of California at Irvine. His research interests are in mobile, wireless and secure networking. Claude Castelluccia hold a MSEE from Florida Atlantic University and a Ph.D. in Computer Science from INRIA Sophia-Antipolis.