IPv6 maintenance Working Group (6man)
INTERNET-DRAFT
Updates RFC 3971 , RFC 3972, RFC 4941 (if approved)
Intended status: Standard Track
Expires: August 25, 2013

H. Rafiee
C. Meinel
Hasso Plattner Institute

February 25, 2013

A Simple Secure Addressing Generation Scheme for IPv6 AutoConfiguration (SSAS)

<draft-rafiee-6man-ssas-02.txt>

Abstract

The default method for IPv6 address generation uses an Organizationally Unique Identifier (OUI) assigned by the IEEE Standards Association and an Extension Identifier assigned to the hardware manufacturer [1] (section 2.5.1 RFC-4291) [RFC4291]. This means that a node will always have the same Interface ID (IID) whenever it connects to a new network. Since the node's IP address does not change, the node is vulnerable to privacy related attacks. To address this problem there are currently two mechanisms being used to randomize the IID that do not use the MAC address or other unique values in the IID generation; Cryptographically Generated Addresses (CGA) [RFC3972] and Privacy Extension [RFC4941]. The problem with the former approach is the computational cost involved for the IID generation and verification. The problem with the latter approach is that it lacks the necessary security and provides the node with only partial protection against privacy related attacks. This document proposes the use of a new algorithm for use in the generation of the IID while, at the same time, securing the node against some types of attack, like IP spoofing. These attacks are prevented with the addition of a signature to the messages sent over the network and by direct use of a public key in the IP address.

Status of this Memo

material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 21, 2013.

Table of Contents

Introduction

IPv6 addresses consist of two parts; the subnet prefix, which is the 64 leftmost bits of the IPv6 address, and the Interface ID (IID), which is the 64 rightmost bits of the IPv6 address. The IEEE Standards Association [1] (section 2.5.1 RFC-4291) [RFC4291] offered a standard for the generation of the IPv6 Interface IDs (IID) which it called the Extended Unique Identifier (EUI-64). EUI-64s are generated by the concatenation of an Organizationally Unique Identifier (OUI) assigned by the IEEE Registration Authority (IEEE RA) with the Extension Identifier assigned by the hardware manufacturer. For example, if a manufacturer's OUI-36 hexadecimal value is 00-5A-D1-02-3, and the manufacture hexadecimal value, for the Extension Identifier for a given component, is 4-42-61-71, then the EUI-64 value generated from these two numbers will be 00-5A-D1-02-34-42-61-71. If the OUI is 24 bits and the extension identifier is also 24 bits (this constitutes the MAC address), then to form the 64-bit EUI address, the OUI portion of the MAC address is inserted into the leftmost 24 bits of the EUI-64 8 byte field and the Extension Identifier is inserted into the rightmost 24 bits of the EUI-64 8 byte field, and then a value of 0xFFFE is inserted between these two 24-bit items. IEEE has chosen 0xFFFE as a reserved value which can only appear in an EUI-64 generated from an EUI-48 MAC address. Then bit 7 (u bit) in the OUI portion of the address should be set. Globally unique addresses assigned by the IEEE set this bit to zero by default indicating global uniqueness.This bit will be set to 1 for locally created addresses, such as those used for virtual interfaces or a MAC address manually configured by an administrator.

There are two mechanisms used to generate a randomized IID that do not make use of a MAC address; CGA [RFC3972] and Privacy Extension [RFC4941]. In this document we discuss the problem inherent with using the current mechanisms and then we explain our solution to the problem, which is to randomize the IID and observing privacy, while, at the same time, providing security to Neighbor Discovery Protocol (NDP) messages, for nodes, in the IP layer. DHCPv6 [RFC3315] can also benefit from this approach for the generation of a random IID or for authentication purposes.

1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document the use of || indicates the concatenation of the values on either side of the sign.

2. Problem Statement

The drawback to using IIDs that do not change over time is one of privacy. The node will generate the same IID whenever it joins a new network thus making it easy for an attacker to track that node when it moves to different networks.

The main problem with the privacy extension mechanism, when using the first approach as explained in section 3.2.1 RFC-4941 [RFC4941], i.e., using stable storage, is the lack of a provision for the use of a security mechanism. The Privacy Extension RFC can partly prevent attacks related to privacy issues, but it cannot prevent attacks related to security issues. For instance, it cannot prevent IP spoofing attacks and it cannot provide proof of the IP address ownership of a node. If one wants to use a secure method, with the privacy extension, then one needs to use CGA. The problem with using CGA is in the computational overhead necessary to compute it when a higher sec value is used and the time that is needed in the verification process. This time is based on the reverse of the steps required to regenerate CGA during the verification process, in addition to the signature verification.

What is clear here is that it is not possible to generate the CGA offline or before hand. This is because the subnet prefix (router prefix) is one of the inputs to the SHA1 algorithm. The other problem with CGA is the apparent lack of a defense against Denial of Service (DoS) types of attack against verifier nodes. In the CGA RFC, there is no explanation as to how to prevent these types of attacks. This means that an attacker can overwhelm the verifier node with false CGA values thus rendering it unable to process further messages.This document also proposes a solution for this type of attack.

To overcome the problem with using the other mechanisms the time needed for IP address generation and verification needs to be reduced. We propose the use of the SSAS algorithm, along with the SSAS signature, to provide a node with the protection it needs to protect it against IP spoofing and spoofing types of attack in the IP layer. Our experimental results [2] show that SSAS is 5 times faster than CGA, when using a sec value of ,0 and 600 times faster than CGA when using the sec value of 1. This will be the same when, in the future, we have faster CPUs because SSAS will also benefit from the future technologies. Currently the generation time for SSAS is less than 1 millisecond so with future new technologies it will be even less.

Note: It is not the intent of this document to obsolete CGA but to propose a simpler and a faster addressing mechanism to use in the randomization of the IID and the for the protection of nodes against the attacks explained below.

2.1. SSAS Applications

2.1.1. Preventing Attacks

The following sections detail some of the attacks that SSAS can prevent.

2.1.1.1. Replay attack

In this type of attack, an attacker might sniff the Neighbor Discovery Protocol enabled networks (NDP) messages and try to copy the legitimate signature and public key to his NDP message and then send this to the sender. But by using the SSAS algorithm, this is prevented with the addition of a timestamp to the NDP message and also with inclusion of this timestamp in the signature. The use of the timestamp works because the timestamp will be valid for a short period of time. (this accounts for clock skews.)

2.1.1.2. IP spoofing

This is a well-known type of attack in NDP. This type of attack is used to attack the Duplicate Address Detection process. In this attack, when a node joins the network and generates a new IP address, the node sends a Neighbor Solicitation (NS) message to check for address collisions in the network. The attacker, in this scenario, spoofs the IP address and responds back to the node with a Neighbor Advertisement (NA) message claiming ownership of this IP address. The SSAS algorithm allows this node to verify other nodes in the network. An attacker does not have the private key for this node, which is needed to generate a SSAS signature, so the verification process will fail.

2.1.1.3. Denial of Service (DoS) attacks

An attacker might send many NDP messages, using invalid signatures, to the victim?s node which then forces the node to busy itself with the verification process. To mitigate this attack, a node SHOULD set a limit on the number of messages (x) that the it can verify, per a certain period of time. Implementations MUST provide a conservative default and SHOULD provide a way for detecting when this limit is reached.

2.1.1.4. Spoofed Redirect Message

Redirect messages, imitating the end host needing redirection, can be sent from any router on the same broadcast segment. The attacker uses the link-local address of the current first-hop router in order to send a Redirect message to a legitimate node. Since that node identifies the message as coming from its first hop router, by use of the link-local address, it accepts the Redirect. The Redirect will remain in effect as long as the attacker responds to the Neighbor Unreachability Detection probes sent to the link-layer address. To preclude this from occurring, the address ownership of the first-hop router should be verified. The use of the SSAS verification process will prevent such an attack.

2.1.2. Nodes with limited resources

SSAS can be used in nodes where limited resources are available for computation. It can provide protection for these nodes against the attacks stated above. Sensor networks are examples of nodes with limited resources (such as battery, CPU, and etc); see RFC-4919 [RFC4919] for the usage of IPv6 in these networks.

Another example could be the use of SSAS in mobile networks during the generation of IP addresses as explained in section 4.4 RFC-6275. The current problem with addressing mechanism in mobile node is that there is no privacy observation as the node usually keeps its Home Address when it moves to another network. If there is a fast secure mechanism, then it is possible set this Home Address and change it and re-register it to the Home network.

3. Algorithm Overview

As explained earlier, one of the problems with the current IID generation approach is the compute intensive processing needed for the IID algorithm generation. Another concern is the lack of security. Since, we assume that a node needs to generate and keep its address for a short time, we tried to keep the IID generation process to a minimum. We also tried to remain within the confines of NDP protocol.

3.1. Interface ID (IID) Generation

To generate the IID, a node needs to execute the following steps.

1. Generate a 16 byte random number called modifier.

2. Generate a 1024-bit key pair (public/private key). These keys SHOULD be stored in a safe place on a local hard disk and the path to this data, and the validation time for these keys, SHOULD be saved in a XML file. It is RECOMMENDED that the public key be generated, on the fly, during the start-up phase of the algorithm generation.

Once a node generates key pairs, it can make use of these keys for a short period of time. It is RECOMMENDED not to use the same keys for more than 10 days in order to prevent the node from being tracked through the use of its public keys. When time expires for the use of these key pairs, the node should generate new key pairs and replace the old one in the XML file. It SHOULD then use the new value for IP address and signature generation.

It is also possible to use ECC [3] with a 192 bit key size. This is equivalent to a1280 bit RSA key size. In this case the packet size would be decreased by a factor 5 times smaller than when using RSA. However, with key sizes 1024 bit and 1280 bit, RSA generation and verification is much faster than ECC. The other problem with the use of ECC is that it could be patented and might not be royalty free.

3. Concatenate the modifier with the timestamp and the public key. The timestamp is a 64-bit unsigned integer field containing a timestamp. The value indicates the number of seconds since January 1, 1970, 00:00 UTC, by using a fixed point format. The format of the timestamp data field is the same as that outlined in section 5.3.1 RFC-3971 [RFC3971].

R1=(modifier(16 bytes)||timestamp(8 bytes)||public key)

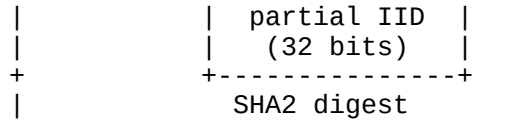
4. Execute SHA2 (256) on the result from step 3.

digest=SHA256(R1)

The use of SHA2 (256) is RECOMMENDED because the chances of finding a collision are less than when using SHA1 and the generation time is acceptable (in microseconds using a standard CPU).

5. Generate a random number between 0 and 20 and call it the start index. This number is used as an index for the SHA2 array of bytes. This value helps randomize the IID and to minimize the chance of a collision in the network. The length of this number is one byte.

6. Take the 32 leftmost bits (starting at the start index) from the resulting output from step 5 (SHA2 digest) and set bits u and g (bits 7 and 8) and call this the partial IID.



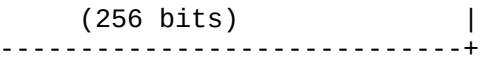


Figure 1 Partial Interface ID

7. Obtain the second byte of the partial IID and call it the start field pubkey. If the value of the start field pubkey is between 0 and the size of public key length, in bytes, minus 4, use this number as an index for the public key array of bytes. Otherwise choose that byte and shift its contents 2 bits to the right (the first two bits will be zero) and set the start field pubkey to this number. This ensures that the value of the start field pubkey will be less than the size of the public key array of bytes, minus 4. This value helps randomize the IID and minimize the chance of a collision in the network. For example, if the second byte of partial IID is 110, the

start field pubkey value will be 110. This value helps randomize the IID and minimize the chance of a collision in the network. For example, if the second byte of the partial IID is 110, then the start field pubkey value will be 110.

If ECC is used for key generation, then the content of the start field pubkey SHOULD be shifted 3 bits to the right. This insures that its value is less than the size of public key array of bytes, minus 4.

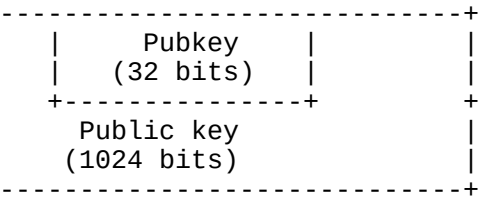


Figure 2 Public key part of Interface ID

8. Concatenate the partial IID with the four bytes from the public key (starting at the start field pubkey) and call this the IID.

Figure 3 Interface ID

9. Concatenate the IID with the local subnet prefix to set the local IP address

10. Concatenate the IID with the router subnet prefix (Global subnet prefix), obtained from the RA message, and set it as a tentative

global IP address. (This IP will be permanent after Duplicate Address Detection (DAD) processing. (for more information about DAD refer to section 4.3.)

3.2. Signature Generation

The SSAS signature is added to NDP messages in order to protect them from IP spoofing and spoofing types of attack. SSAS will prove IP address ownership, as does the CGA generation algorithm, but using fewer steps. To generate the SSAS signature, the node needs to execute the following steps:

1. Concatenate the timestamp with the 16 byte public key (that starts at the start field pubkey) (see figure 4) and the global IP address. The start field pubkey is one of the numbers that was introduced in step 7 of section 4.1.

2. Sign the resulting value from step 1, using the RSA private key unless we use ECC, and call the resulting output the SSAS signature.

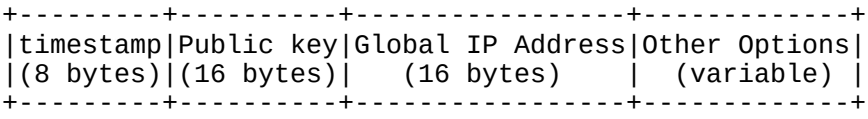


Figure 4 SSAS Signature

If NDP messages contain other data that must be protected, such as important routing information, this data SHOULD also be included in the signature. The signature is designed for the inclusion of any data needing protection. If there is no data that needs protection, then the signature will only contain the timestamp, 16 byte public key and Global IP address (Router subnet prefix plus IID).

3.3. Generation of NDP Messages

After a node generates its IP address, it should then process Duplicate Address Detection in order to avoid address collisions in the network. To do this, the node generates a Neighbor Solicitation (NS) message. The format of a NS message is shown in figure 5. The SSAS signature is added to the ICMPv6 options of NS messages. The SSAS signature data field is an extended version of the standard format of the RSA signature option of SEND [RFC3971]. The timestamp option is the same as that used with SEND. In the SSAS signature, the data field contains type, length, reserved, Other Len, pubkey len, public key, SSAS signature, and padding.

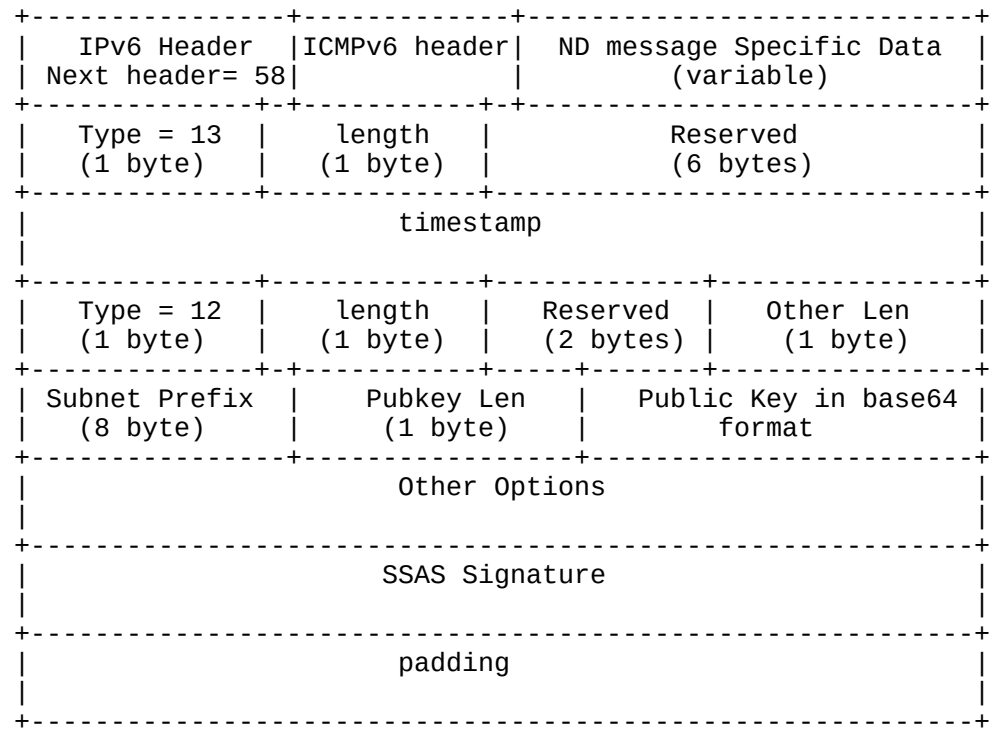


Figure 5 NDP Message Format with SSAS Signature Data Field

This document proposes an update to the SEND RFC in order to replace the RSA signature field with the SSAS signature data field and to add SSAS as a new option to SEND messages.

3.3.1. SSAS signature data field

- Type: This option should be set to 12.

- Length: The length of the Signature Data field, including the Type, Length, Reserved, pubkey Len, public key, Signature and padding, should be a multiple of eight.

- Reserved: A 2 byte field reserved for future use. The value MUST be initialized to zero by the sender, and MUST be ignored by the receiver.

- Other Len: The length of other options in multiples of eight. The length of this is 1 byte.

- Subnet Prefix: This is the router subnet prefix.

- PubKey Len. The length of the public key in multiples of eight.

- Public key. Base64 format of the public key

- Other Options. This variable-length field contains important data that needs to be protected in the packet . The padding would be added, as many bytes long as remain after the end of the field, if the Other options is not a multiple of eight.

- Padding. This variable-length field contains padding, as many bytes long as remain after the end of the signature, if the signature is not a multiple of eight.

All NDP messages should contain the SSAS signature data field which allows receivers to verify senders. If a node receives a solicited NA message in response to its NS message showing that another node claims to own this address, then, after a successful verification process, this node increments the modifier by one and again repeats steps 3 thru 8 of section 4.1 . If, for a second time, the node receives the same claim, then it considers it an attack and will use that IP address.

3.4. SSAS verification process

A node’s verification process should start when it receives NDP messages.

Following are the verification steps:

1. Obtain the timestamp from the NDP message and call this value t1.

2. Obtain the timestamp from the node’s system, convert it to UTC, and call this value t2.

3. If (t2- x) < = t1 < = (t2 + x) go to stop 4. Otherwise, the message SHOULD be discarded without further processing. (The value of x is dependent on network delays and network policy. One might choose10 minutes (600 seconds) as a flexible way of handling network delays.)

4. Obtain the public key from the SSAS signature data field.

5. Compare this to its own public key. If it is not the same, go to the next step. Otherwise, the message should be discarded without further processing.

6. Obtain the second byte of the partial IID and call it the start field pubkey. If the value of the start field pubkey is between 0 and the size of public key length, in bytes, minus 4, use this number as an index for the public key array of bytes. Otherwise choose that byte and shift its contents 2 bits to the right (the first two bits

will be zero) and consider this number the starting index of the public key array of bytes. This ensures that the value of that byte will be less than the size of the public key array of bytes, minus 4. Set the start field pubkey to this number.

If ECC is used for key generation, then the content of the start field pubkey SHOULD be shifted 3 bits to the right. This insures that its value is less than the size of public key array of bytes, minus 4.

7. Obtain the IID from the sender?s source IP address. (64 rightmost bits of the IPv6 address)

8. Compare the 32 leftmost bits, starting at the start field pubkey of the public key, to the 32 rightmost bits of the IID of the sender?s IP address. If they are the same, go to the next step. Otherwise, the message should be discarded without further processing

9. Obtain the subnet prefix from the SSAS signature data field.

10. Concatenate the timestamp with the 16 bytes of the public key, (starting from start field pubkey), the subnet prefix, the sender?s IID, and other options (if any) and call this entity the plain message.

11. Obtain the SSAS signature from the SSAS signature data field.

12. Verify the Signature using the public key, and then enter the plain message and the SSAS signature as an input to the verification function. If the verification process is successful, process the message. Otherwise, the message should be discarded without further processing.

4. Security Considerations

As a security consideration what one might ask is what are the odds of an attacker being able to generate a public key having four sequential bytes the same as the last rightmost 32 bits of the IID If he could, he could then generate the signature using his own private key and thus break the SSAS.

Mathematically it has been shown that the probability of matching 32 bits in the public key against 32 bits in the IID is about

pow(1/2,32) where pow is the power function, 2 is a base and 32 is a exponent. Since the use of a public key and IP address with a maximum lifetime of 10 days is RECOMMENDED, the probability of an attacker finding the same value is 0.0008, a very small value. When one also considers the probability of an attacker being able to generate a public key whose 32 bits, starting from an arbitrary point, matches the 32 bits of the public key generated using the SSAS algorithm, then the probability of his success is diminished even further. This shows the strength of this algorithm against brute force attacks while, at the same time, by using the signature and finding a binding between the IP address and the public key, it provides proof of IP address ownership at a speed that is about 600 times faster than that of the CGA algorithm [2]. (based on the implementation results, the average time to generate SSAS is 882.77 microseconds).

Another consideration concerns Routers wanting to use this algorithm in place of CGA. As explained in RFC SEND, for routers, the use of a Trusted Authority is RECOMMENDED along with verifying router certificates using these third parties. This will prevent a node from claiming to be a router. But for nodes, rather than routers, SSAS can provide protection against the types of attacks explained above.

5. IANA Considerations

This document defines a new algorithm for the generation of an Interface ID in IPv6 networks.

6. Conclusions

Privacy has become a very important issue in recent years. A solution for preventing a node from being tracked by an attacker is to change the node’s IP address frequently and by generating a random IID each time a node wants to generate a new IP address. There are two solutions available for randomizing the IID; CGA and Privacy Extension. The former algorithm is compute intensive and the latter algorithm is lacking in security. This document introduced a new algorithm as a solution for providing privacy by randomizing the IID and for providing security with the addition of a SSAS signature to the NDP message and finding a binding between the public key and the IP address. Our experimental results [2] show a definite improvement in the computation time for the SSAS algorithm as compared to that for the CGA algorithm. We also note that the probability of having collisions with IP addresses, when using the SHA2 digest and the public key, with a randomized 62 bit selection, approximates pow(1/2,62) where pow is the power function, 2 is a base and 62 is a

exponent (u and g bits are ignored) . Moreover, the probability of an attacker finding the public key which matches 32 rightmost bits of the IID within 10 days approximates 0.0008. This means this algorithm is secure enough for wide usage.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4291] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture," RFC 4291, February 2006.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)," RFC 3972, March 2005.

[RFC4941] Narten, T., Draves, R., Krishnan, S., "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and Nikander, P., "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M. , " Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC4919] Kushalnagar, N., Montenegro, G., Schumacher, C.," IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

7.2. Informative References

[1] IEEE Standards Association, http://standards.ieee.org/develop/regauth/tut/eui64.pdf, 2012

[2] Rafiee, H., "Research Results", http://ipv6sra.rozanak.com/Jan2013_CGA_SSAS_Comparison.pdf, 2013

[3] Brown, R., L., D. : SEC 1: Elliptic Curve Cryptography, Certicom Research, http://www.secg.org/download/aid-780/sec1-v2.pdf, 2009

Authors' Addresses

Hosnieh Rafiee
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany
Phone: +49 (0)331-5509-546
Email: ietf@rozanak.com

Dr. Christoph Meinel
(Professor)
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany
Email: meinel@hpi.uni-potsdam.de