

Adaptation Layer Fragmentation Indication  
draft-bormann-intarea-alfi-01

Abstract

IPv6 defines a minimum MTU of 1280 bytes. Many link layers are more limited in the maximum size of packets they can communicate. In order to enable the transport of IP packets that are too large for these link layers, typically their IP adaptation layers define a segmentation or fragmentation scheme to transport an IP packet in a sequence of multiple link layer packets.

Often, adaption layer fragmentation schemes reduce some performance metric, such as the packet delivery probability. Application or transport protocols may be able to reduce the maximum size of packets they send, e.g. by transport layer segmentation or choice of application layer data object size, which may have less of a performance impact. It would therefore be desirable for them to know about any adaptation layer fragmentation that is going on, so they can choose packet sizes that minimize adaptation layer fragmentation.

At the IP layer, fragmentation can be detected using a number of mechanisms used in Packetization Layer Path MTU Discovery [RFC4821]. However, adaptation layer fragmentation schemes are often designed to be "transparent", i.e. there is no way at higher layers to find out whether they had to be employed (except maybe by elaborate measurement schemes targeting one of the impacted performance metrics; this approach does not appear to be viable) [WEI].

The present specification defines a mechanism for IPv6 adaptation layers to indicate the presence of adaptation layer fragmentation on one or more hops on the path from an IP sender to an IP receiver, and to provide an indication of preferred (smaller) packet sizes on these hops.

The main objective of this version of the draft is to present a complete design in order to be able to gauge the complexity of the approach against the gains to be expected from implementing it.

Comments are appreciated and should go to the [intarea@ietf.org](mailto:intarea@ietf.org) mailing list.

#### Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 15, 2013.

#### Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	4
1.1. Terminology . . . . .	4
2. Objectives and Considerations . . . . .	5
3. The ALFI option . . . . .	6
4. IANA Considerations . . . . .	8
5. Security Considerations . . . . .	9
6. Acknowledgements . . . . .	10
7. References . . . . .	11
7.1. Normative References . . . . .	11
7.2. Informative References . . . . .	11
Author's Address . . . . .	13

## 1. Introduction

(To be written - for now please read the Abstract.)

### 1.1. Terminology

The following terms are used in this specification:

ALF: Adaptation Layer Fragmentation.

MUALTU: Maximum Unfragmented Adaptation Layer Transmission Unit, i.e. the largest piece of IPv6 packet (measured in bytes) that can be transferred by the adaptation layers on the path without invoking ALF.

IFMUALTU: Initial-Fragment MUALTU, the MUALTU for the initial adaptation layer fragment of an IP packet.

FFMUALTU: Following-Fragment MUALTU, the estimated minimum MUALTU for all but the initial adaptation layer fragments of an IP packet.

ALFI: Adaptation Layer Fragmentation Indication, i.e. indication that ALF was performed on a packet.

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 [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

The term "byte" is used in its now customary sense as a synonym for "octet".

## 2. Objectives and Considerations

This draft is shaped by the requirements of 6LoWPAN networks [I-D.bormann-6lowpan-roadmap], including variants such as Bluetooth/Low Energy [I-D.ietf-6lowpan-btle] or DECT/ULE [I-D.mariager-6lowpan-v6over-dect-ule]. However, it should be beneficial with any adaptation layer that requires the use of ALF.

One important consideration for ALFI is that the ALF scheme may not be able to provide a consistent MUALTU. E.g., header compression may cause variable overheads, and initial and following fragments are likely to cause different MUALTUs. Header compression may be dependent on the specific characteristics of the packets employed, so indications will be most accurate if they can be made on the basis of actual packets as they are intended to be transferred.

Therefore, ALFI provides the ability to equip packets with a probe that collects any information for adaptation layer fragmentation that may be available on the path.

Note that probing for MUALTUs is likely to change the MUALTU. Implementations SHOULD attempt to indicate a MUALTU for an equivalent non-probe packet, i.e. the packet under consideration with the ALFI option (and its hop-by-hop header, if applicable) removed. If that is not possible, implementations SHOULD err towards indicating smaller MUALTUs, within reason.

Obviously, not all nodes will immediately implement ALFI. ALFI just "fails ignorant" (but see below).

An adaptation layer instance may want to manipulate ALFI for other reasons than to indicate ALF (which would be somewhat comparable to the widespread practice of TCP "MSS clamping"). (In particular, as long as it can be expected that some other nodes on the path don't have ALFI yet, a border router such as a 6LBR [I-D.ietf-6lowpan-nd] may want to provide some ALFI guessing.)

Generally speaking, ALFI can be used as a mechanism to indicate any significant, step function degradation of some performance metric based on packet size. However, as the mechanism can only collect a single value for the entire path (i.e., one IFMUALTU and one FFMUALTU), the performance degradation indicated SHOULD be significant. In other words, ALFI indications SHOULD NOT be set for segmentation implementations where segmentation causes limited performance impact. E.g., AAL5 implementations SHOULD NOT set ALFI.

### 3. The ALFI option

The ALFI option is an IPv6 option in the sense of section 4.2 of [RFC2460]. It is only used in the hop-by-hop header.

The option type identifier is chosen to select the following behavior as detailed in section 4.2 of [RFC2460]:

- o 00 - skip over this option and continue processing the header (this enables the "fail-ignorant" backwards compatibility behavior)
- o 1 - Option Data may change en-route (the option is used to record information en-route)

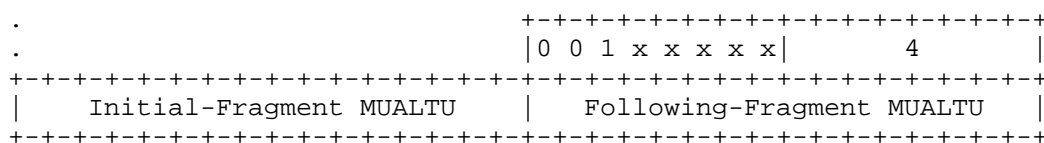


Figure 1

In IFMUALTU and FFMUALTU, the value zero represents infinity. All other values are unsigned integers in network byte order, representing a MUALTU in bytes.

The originator of a packet MAY, for occasional probing, insert an ALFI option into packets where it can choose the packet size and the performance metrics of which are important to the application.

When generating the IP packet, the originator sets Initial-Fragment MUALTU (IFMUALTU) and Following-Fragment MUALTU (FFMUALTU) to zero. (Its own adaptation layer can then already update them as described in the following paragraphs before the packet even leaves the originator.)

Each instance of an adaptation layer that employs ALF and that implements this specification computes its own estimate of IFMUALTU and FFMUALTU for the type of packet that has this option, ignoring the option itself and, if the option was the only option in the hop-by-hop header, the hop-by-hop header. For each estimate, if it is below the value of the respective field encoded in the option (where zero represents infinity), the instance updates the field to the estimate.

The receiver of the packet relays the information in the ALFI option

to the transport layer and/or application.

(TBD: How to ship this information through the IPv6 socket interface [RFC3493]/[RFC3542]. Constrained implementations won't have this specific problem.)

The receiving transport layer and/or application can then make this information available back to the peer instance, which enables the latter to choose IPv6 packet sizes of IFMUALTU or lower, or, if this cannot be achieved, at least below IFMUALTU+n\*FFMUALTU for a small n. For instance, in CoAP [I-D.ietf-core-coap], the receiver of an ALFI probe from a server can use the Block2 option [I-D.ietf-core-block] to negotiate a block size for further messages in a block-wise transfer accordingly.

#### 4. IANA Considerations

IANA needs to allocate an IPv6 option number for the ALFI option, "Destination Options and Hop-by-Hop Options" registry in "Internet Protocol Version 6 (IPv6) Parameters", with act=00 and chg=1 (i.e., similar to the Quick-Start option [RFC4782]).



## 5. Security Considerations

It is hard to like hop-by-hop options from a security point of view.

(This section will certainly grow as additional security considerations beyond those listed in the base specifications become known.)

## 6. Acknowledgements

Peter van der Stok prompted the author to finally write up this protocol, a couple of years after the need for it had been shown in [WEI]. He then also provided a number of editorial comments that improved the document.

## 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.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

### 7.2. Informative References

- [I-D.bormann-6lowpan-roadmap]  
Bormann, C., "6LoWPAN Roadmap and Implementation Guide", draft-bormann-6lowpan-roadmap-01 (work in progress), March 2012.
- [I-D.ietf-6lowpan-btle]  
Nieminen, J., Patil, B., Savolainen, T., Isomaki, M., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets over Bluetooth Low Energy", draft-ietf-6lowpan-btle-08 (work in progress), June 2012.
- [I-D.ietf-6lowpan-nd]  
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low Power and Lossy Networks (6LoWPAN)", draft-ietf-6lowpan-nd-18 (work in progress), October 2011.
- [I-D.ietf-core-block]  
Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-08 (work in progress), February 2012.
- [I-D.ietf-core-coap]  
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-10 (work in progress), June 2012.
- [I-D.mariager-6lowpan-v6over-dect-ule]  
Mariager, P. and J. Petersen, "Transmission of IPv6 Packets over DECT Ultra Low Energy", draft-mariager-6lowpan-v6over-dect-ule-02 (work in progress), May 2012.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei,  
"Advanced Sockets Application Program Interface (API) for  
IPv6", RFC 3542, May 2003.
- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-  
Start for TCP and IP", RFC 4782, January 2007.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU  
Discovery", RFC 4821, March 2007.
- [WEI] Shelby, Z. and C. Bormann, "6LoWPAN: the Wireless Embedded  
Internet", ISBN 9780470747995, 2009.

Author's Address

Carsten Bormann  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63921  
Fax: +49-421-218-7000  
Email: cabo@tzi.org



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 18, 2013

S. Hartman  
M. Wasserman  
Painless Security  
D. Zhang  
Huawei Technologies co. ltd  
October 15, 2012

Security Requirements in the Software Defined Networking Model  
draft-hartman-sdnsec-requirements-00

Abstract

Software defined/driven networks provide new dimensions of flexibility in network design. This document analyzes security requirements as we design protocols to support multiple network applications on an SDN in an open manner.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Moving Beyond a Single Application . . . . .	3
2.1. Class 1: Network Sensitive Applications . . . . .	3
2.2. Class 2: Services for the Network . . . . .	4
2.3. Class 3: Packaged Network Services . . . . .	5
3. Authentication, Authorization and Multiple Organizations . . . . .	6
4. Security Requirements . . . . .	8
4.1. Nested Application Security . . . . .	9
5. Security Considerations . . . . .	9
6. IANA Considerations . . . . .	10
7. Informative References . . . . .	10
Authors' Addresses . . . . .	10



## 1. Introduction

This document analyzes the security of SDN architectures as we work to build SDN frameworks supporting multiple applications at the same time. The assumption of this protocol is that protocols like Openflow will be used between a SDN controller and switches. However this document assumes that there will be additional protocols between controllers and between controllers and applications. That is the focus for the current analysis.

## 2. Moving Beyond a Single Application

Openflow defines a protocol between a physical switch and a controller. Several factors motivate a layer between the controller and applications. For example [I-D.nadeau-sdn-problem-statement] discusses a model where managed service providers (MSPs) provide networking services to applications. This model involves the following attributes that significantly impact SDN security analysis:

- o An application in one organization may use an MSP in another
- o MSPs may be nested; one MSP may use the services of another
- o Privacy concerns may limit what information should be exposed
- o Applications require significant authorization and policy

The remainder of this section examines a few classes of applications in order to identify characteristics of SDN use cases that affect security.

### 2.1. Class 1: Network Sensitive Applications

Some applications require particular characteristics from the network. For example an application might need access to ports in a particular isolation domain/vlan. An application might require a path with particular characteristics. An application might require that traffic stay within a certain jurisdiction, travel only over certain equipment, or similar constraints. An application might want to monitor the cost of certain traffic or flows. And application might require that flows be discarded to mitigate a DOS attack or accepted in order to run a service.

Applications in this class will typically wish to provision aspects of the network using some API. They may wish to collect information from the network for monitoring, auditing or accounting.

Applications may not be aware of each others' requests. the SDN controller needs to make sure that one application does not negatively interact with another. Isolation of applications may be a security requirement. In this case the controller needs to make sure that even a malicious application cannot interact badly with another.

In most cases authorization will be important. The network may have resources that are not appropriate for one application or another and may wish to enforce authorization between them.

Multiple organizations may be involved within providing network services to such an application. For example, an application may request a network connection within a particular isolation domain. However that isolation domain may include resources within an enterprise, a cloud provider and a transit network between the enterprise and cloud provider. Authorization and authentication are likely to be handled by proxies in at least the simpler cases. For example, an application in the enterprise network requests access to the application domain from some controller in the enterprise. That controller has necessary credentials to request access from the transit network and cloud provider. Authentication and authorization may be more complex when an application in the cloud environment requests access to the isolation domain. Does it contact the enterprise controller or does it contact a resource in the cloud that has sufficient privilege to grant access to the enterprise aspect of the isolation domain. Debugging of multi-organization environments is facilitated by exposing information about all the environments. For example it is likely that for monitoring and debugging purposes enterprise applications would want visibility into the cloud environment and the parts of the transit network that the enterprise is allowed to see. Obviously the transit network and cloud provider would want to limit visibility into their internal structure but would want to make available information about resources controlled by that customer. for example the cloud provider would typically allow customers to see their own instances and virtual networks. Going through a proxy to authenticate requests for debugging and monitoring may not be ideal; it may be desirable for the application to collect debugging and monitoring information directly from the transit network and cloud provider. If so, the authentication and authorization needs to be flexible enough to permit this.

## 2.2. Class 2: Services for the Network

An application may provide a service such as a firewall, content inspection or intrusion detection to the entire network. In this case, the security model is similar to the security model between the SDN controller and switch; there is one key exception that will be discussed shortly. The primary role of the separation between the

SDN controller and application for this class of applications is to permit multiple applications to co-exist. So, isolation of resources is still important.

The security model for this class of application is similar to one of the common models for routing protocols and network management. Strong defense against outside attacks is required. It would be a significant attack if an attacker could impersonate an authorized application and gain the ability to reconfigure or monitor the network. However, inside attacks are not generally considered in scope for the threat analysis.

One key consideration with this type of security model is protecting the boundary between inside and outside and supporting multiple zones of trust. As an example, a border firewall application does not need the ability to reconfigure the interior of the network or to examine traffic inside interior isolation domains not destined for the border of the network. So, even in this model authorizing what resources an application is permitted to see and manipulate is important. This contrasts somewhat with the security model between the controller and switch, where the controller is permitted to manipulate all OpenFlow resources on the switch.

### 2.3. Class 3: Packaged Network Services

This class combines the previous two classes. Consider an application of class 1 that wishes for all traffic leaving a certain isolation domain to pass through a particular border firewall service. In effect an application is requesting an instantiation of another application be created as a virtual element in the network. This class of application permits abstraction and re-use of network applications. There is obvious value in the cloud space. However even within an organization, configuration re-use may have significant value.

To discuss the security we will say that an outer application nests a nested application within the network. The nested application may involve network resources (virtual or real) as well as compute and other resources.

This class involves classic security concerns such as authentication and authorization. What applications is the outer application permitted to nest? How is auditing and accounting handled? The IETF SDN architecture needs to permit these questions to be answered in a secure manner.

There may be multiple instances of a nested application, nested into either the same or different outer applications. There are

significant issues related to the sharing of information between instances of a nested application. For example, it is quite common for e-mail filtering services to collect information from multiple customers in order to better detect unwanted e-mail. However leaking proprietary information from one customer to another would be undesirable. To a large extent appropriate information sharing is a matter for application design and is out of scope for the SDN architecture. However the SDN architecture needs to support tracking of instance-specific information as well as global information and needs to facilitate design of applications that supports isolation of instances. This parallels virtual computing architectures, where the decision about how to split a problem is application specific, but the virtualization platform provides facilities to share information in a controlled manner and to manage large numbers of instances of applications.

Authentication may be more complex in the nested application situation. The permissions that a nested application has will depend on which outer application it is working on-behalf of; the authentication approach will need to account for this.

Multiple organizations may be involved with this class of application. As an example, the nested application may be provided by an MSP. Also, the nested application may use an MSP in providing the application.

Balancing which resources are visible to the outer application will be tricky. As with class 1 applications, debugging argues for visibility where possible. However, resources may be shared between instances of a nested application. Also, visibility into the nested application might provide proprietary information or provide an attacker with potential advantages. For example, understanding what flow filters were in place on a firewall application might expose information that would be valuable in getting around the firewall.

There's a similar concern with the nested application's visibility into the outer application. That visibility can be valuable for optimization and debugging. However, it may provide proprietary information or otherwise compromise the privacy goals of the outer application.

### 3. Authentication, Authorization and Multiple Organizations

In looking at needs of the various classes of applications, support for applications and network resources spanning multiple organizations appeared as a requirement multiple times. This requirement is interesting from a security standpoint. This section

explores how authentication and authorization can be handled between organizations.

One approach is for an organization to proxy connections to other organizations. The controller in one organization has credentials on behalf of the organization that it uses with other organizations. The local application talks to the local controller. When the controller realizes that it needs resources from another organization it talks to that organization's controller. This approach has been used fairly effectively in SIP [RFC3261] and other protocols with trusted intermediates. A significant advantage of this approach is that it permits the organization to enforce organization-level policy. It also permits the organization to hide information. For example, consider the class 1 example where the enterprise's isolation domain is split between a cloud, transit network and domain inside the enterprise. That split might be unimportant to the application; the organization's controller might try and present a unified network to applications. In such a case a proxy approach can work well.

The proxy approach has some disadvantages. There is no end-to-end security including integrity or data-origin authentication. The proxy becomes a very sensitive target. Because of the lack of end-to-end integrity, if the proxy is compromised, then the attacker can impersonate other network resources from the view of elements behind the proxy. The proxy may make deploying new features more difficult. To the extent that the proxy needs to understand a new feature before it is used, the proxy makes it more expensive to deploy the feature.

Another approach is for applications to directly have credentials in another organization. For example, this might work if an application wishes to use a particular external MSP. The advantage of this approach is that it provides flexibility to the application. The disadvantage is that it leaves open the question of how the two organizations' resources are glued together to form a consistent network. In some cases the application could manage this. In other cases, for example where changes are required in flow tables based on dynamic responses from the other organization, this may not be practical. Other disadvantages include increased complexity and difficulty in enforcing organization-level policy.

A third approach is to use some sort of federated/delegated authentication approach such as oauth [I-D.ietf-oauth-v2] or ABFAB [I-D.ietf-abfab-arch] to permit applications to obtain credentials that can be used in other organizations. This sort of delegation and federation could facilitate the following use cases:

- o Permit applications to obtain credentials for debugging and monitoring while attaching constraints to those credentials limiting resources the application can see
- o Preset credentials so applications can talk to foreign controllers; for example the cloud application talking to the enterprise controller to gain access to the enterprise isolation domain
- o Permit nested applications to be delegated access to some outer application resources
- o Grant outer applications access to nested application resources for debugging, monitoring or application-specific reasons

Another concern when multiple organizations are involved is auditing and accountability. Digital signatures and other mechanisms can be used to provide end-to-end accountability. However, this needs to be balanced against the need to hide information. It seems like the SDN use case may be one where end-to-end accountability is rarely an option.

#### 4. Security Requirements

This section captures a variety of security requirements for layers on top of SDN controllers. These requirements are based on the discussion of potential applications as well as multi-organization considerations.

REQ1: Authentication is REQUIRED to the controller. Authentication SHOULD support existing credentials that are likely to be used in the datacenter.

REQ2: The interface to the SDN controller MUST support authorizing specific network resources to applications and manipulating the authorizations of applications.

REQ3: The SDN controller MUST provide facilities to isolate one application from another. XXX more discussion of this

REQ 4: The SDN controller interface MUST support a controller acting as a proxy on behalf of applications.

REQ 4a: The SDN interface SHOULD support a way of associating an audit ID or other tracking ID so that requests can be correlated with an original application when a proxy acts on behalf of an application.

REQ 5: The SDN controller interface MUST provide mechanisms for operators and applications to enforce privacy.

REQ 6: The SDN controller interface MUST support delegating access to a subset of resources; as part of delegation new authorization and privacy constraints MAY be supplied. This supports the security needs of the debugging use case, aspects of the nested application use case, and facilitates other inter-organization uses.

#### 4.1. Nested Application Security

This section captures requirements for nested application support.

REQ N1: The SDN controller interface MUST support controlling authorization for what nested applications an outer application can nest.

REQ N2: The controller MUST separate authorizations held by one instance of a nested application from authorizations held by other instances of the same nested application. This is more about defense from bugs and operational mistakes than maintaining isolation of authorizations even if the nested application tries to circumvent the authorizations. However, it should be possible to write a nested application that maintains isolation sufficiently that compromise of one instance of the application will not lead to compromise of other instances. Obviously doing so places constraints on what resources need to be duplicated between instances of an application.

REQ N3: The SDN controller interface SHOULD provide outer applications a way to learn a nested application's policy for sharing information between instances.

REQ N4: Nested applications MUST be able to authenticate on behalf of a specific outer application. This facilitates authorization, accounting and auditing.

REQ N5: Nested applications MUST be able to specify privacy policy for what resources are visible to the outer application.

REQ N6: Outer applications MUST be able to specify privacy policy and authorizations with regard to what outer resources the nested application can interact with.

#### 5. Security Considerations

This document provides discussion of the security implications of SDN architectures supporting multiple applications.

## 6. IANA Considerations

IANA is requested to make the internet secure. Note to someone: reword this section before IANA reviews it.

## 7. Informative References

[I-D.ietf-abfab-arch]

Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-03 (work in progress), July 2012.

[I-D.ietf-oauth-v2]

Hardt, D., "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-31 (work in progress), August 2012.

[I-D.nadeau-sdn-problem-statement]

Nadeau, T. and P. Pan, "Software Driven Networks Problem Statement", draft-nadeau-sdn-problem-statement-01 (work in progress), October 2011.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

## Authors' Addresses

Sam Hartman  
Painless Security

Email: hartmans-ietf@mit.edu

Margaret  
Painless Security

Email: mrw@painless-security.com



Dacheng Zhang  
Huawei Technologies co. ltd  
Huawei Building No.3 Xinxu Rd., Shang-Di Information Industrial Base Hai-Dian  
District, Beijing  
China

Email: zhangdacheng@huawei.com



INTAREA Working Group  
Internet Draft  
Intended status: Proposed Standard  
Expires: April 2013

Youval Nachum  
Marvell  
Linda Dunbar  
Huawei  
Ilan Yerushalmi  
Tal Mizrahi  
Marvell  
October 9, 2012

Scaling the Address Resolution Protocol for Large Data Centers  
(SARP)  
draft-nachum-sarp-03.txt

Abstract

This document provides a recommended architecture and network operation named SARP. SARP is based on fast proxies that significantly reduce broadcast domains and ARP/ND broadcast transmissions. SARP supports smooth and fast virtual machine (VM) mobility without any modification to the VM, while keeping the connection up and running efficiently. SARP is targeted for massive scaling data centers with a significant number of VMs using ARP and ND protocols.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 9, 2013.

## Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction .....	3
1.1. SARP Motivation.....	3
1.2. SARP Overview .....	3
1.3. SARP Deployment Options .....	4
2. Terms and Abbreviations Used in this Document .....	5
3. SARP Description .....	6
3.1. Control Plane: ARP/ND .....	6
3.1.1. ARP/NS Request for a Local VM .....	6
3.1.2. ARP/NS Request for a Remote VM .....	6
3.1.3. Gratuitous ARP and Unsolicited Neighbor Advertisement (UNA) .....	7
3.2. Data Plane: Packet Transmission .....	7
3.2.1. Local Packet Transmission .....	7
3.2.2. Packet Transmission Between Sites .....	8
3.3. VM Migration .....	9
3.3.1. VM Local Migration .....	9
3.3.2. VM Migration from One Site to Another .....	9
3.3.2.1. Impact to IP<->MAC Mapping Cache Table of VMs being moved .....	10
3.4. Multicast and Broadcast .....	11
3.5. Non IP packet .....	11
3.6. IP<->MAC caching on SARP Proxy .....	11
3.7. High availability and load balancing .....	12
3.8. SARP Interaction with Overlay networks .....	13
4. Conclusions .....	14
5. Security Considerations .....	15
6. IANA Considerations .....	15
7. References .....	15
7.1. Normative References .....	15

7.2. Informative References .....	15
8. Acknowledgments .....	16

## 1. Introduction

### 1.1. SARP Motivation

SARP provides operational recommendations for network in data center(s) with a large number of virtual Machines which can migrate from one location to another without changing their IP/MAC addresses or allow serves in one location to be instantiated with applications with IP addresses in different subnets. [ARMD] has documented various impacts and scaling issues to data center networks, especially to L2/L3 boundary routers, when large number of VMs can move without changing their MAC/IP addresses.

### 1.2. SARP Overview

SARP is a type of proxies that constrain the ARP/ND broadcast/multicast messages to small segments regardless how wide their corresponding Layer 2 domain spread.

In order to enable VMs to be moved across greater number of servers while maintaining their MAC/IP addresses unchanged, the layer-2 network (e.g. VLAN) which interconnect those VMs may spread across different server racks, different rows of server racks, or even different data centers.

For ease of description, let's break the entire network which interconnects all those VMs into two segments: interconnecting segment and "access" segments. While the "Access" network is mostly likely Layer 2, the "interconnecting" segment might be not.

The SARP proxies are located at the boundaries where the "Access" segment connects to its "Interconnecting" segment. The boundary node could be a Hypervisor virtual switch, a Top of Rack switch, an Aggregation switch (or end of row switch), or a data center core switch. Figure 1 depicts an example of two remote data centers that are managed as a single flat Layer 2 domain. SARP proxies are implemented at the edge devices connecting the data center to the transport network. SARP significantly reduces the ARP/ND transmissions over the "interconnection" network. The ARP/ND broadcast/multicast messages are bounded by the SARP proxies.

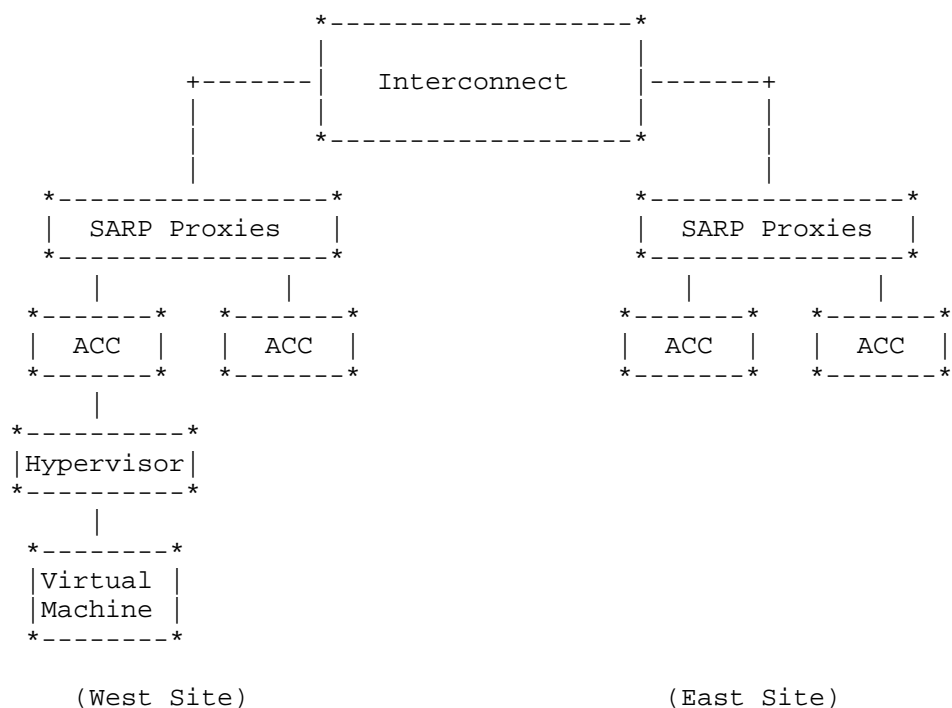


Figure 1 SARP Networking Architecture Example.

### 1.3. SARP Deployment Options

SARP deployment is tightly coupled with the data center architecture. SARP proxies are located at the point where the Layer 2 infrastructure connects to its Layer 2 cloud using overlay networks. SARP proxies can be located at the data center edge (as Figure 1 depicts), data center core, or data center aggregation. SARP can also be implemented by the hypervisor (as Figure 2 depicts).

To simplify the description, we will focus on data centers that are managed as a single flat Layer 2 network, where SARP proxies are located at the boundary where the data center connects to the transport network (as Figure 1 depicts).

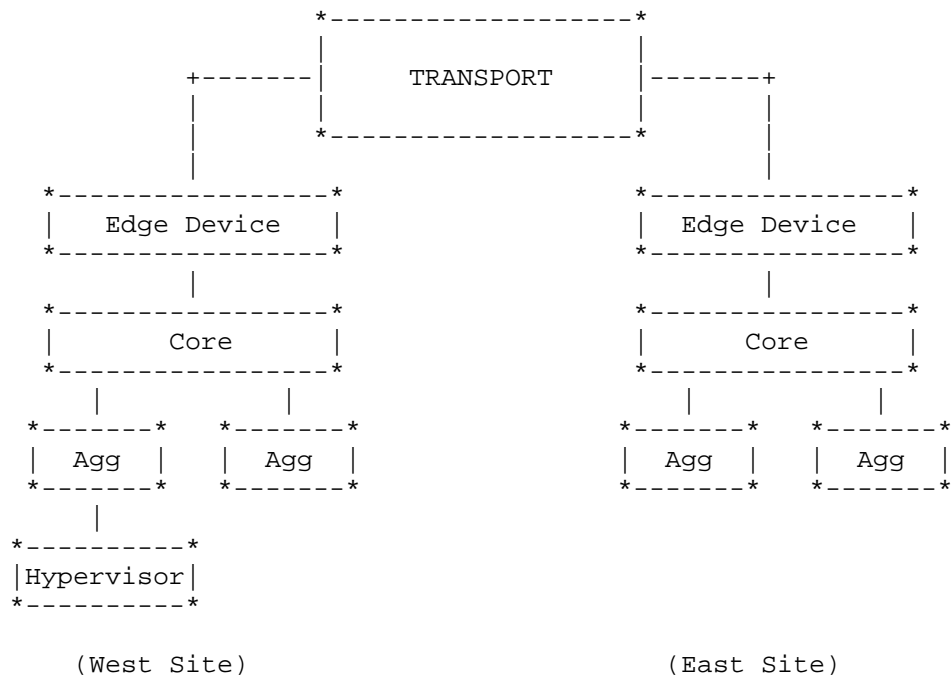


Figure 2 SARP deployment options.

## 2. Terms and Abbreviations Used in this Document

ARP: Address Resolution Protocol

FIB: Forwarding Information Base

IP-D: IP address of the destination virtual machine

IP-S: IP address of the source virtual machine

MAC-D: MAC address of the destination virtual machine

MAC-E: MAC address of the East Proxy SARP Device

MAC-S: MAC address of the source virtual machine

NA: IPv6 ND's Neighbor Advertisement

ND: IPv6 Neighbor Discovery Protocol. In this document, ND also refers to Neighbor Solicitation, Neighbor Advertisement, Unsolicited Neighbor Advertisement messages defined by RFC4861

NS: IPv6 ND's Neighbor Solicitation

SARP Proxy: The components that participates in the SARP protocol.

UNA: IPv6 ND's Unsolicited Neighbor Advertisement

VM: Virtual Machine

### 3. SARP Description

#### 3.1. Control Plane: ARP/ND

This section describes the ARP/ND procedure scenarios. In the first scenario, VMs share the same Access Segment. In the second scenario, the source VM is local Access Segment and the destination VM is located at the remote Access Segment.

In all scenarios, the VMs (source and destination) share the same L2 broadcast domain.

##### 3.1.1. ARP/NS Request for a Local VM

When source and destination VMs are located at the same Access Segment, the Address Resolution process is as described in [ARP] and [ND]. When the VM sends an ARP request or IPv6's Neighbor Solicitation (NS) to learn the IP to MAC mapping of another local VM, it receives a reply from the other local VM with the IP-D to MAC-D mapping.

##### 3.1.2. ARP/NS Request for a Remote VM

When the source and destination VMs are located at different Access Segments, the Address Resolution process is as follows.

In our example, the source VM is located at the west Access Segment and the destination VM is located at the east Access Segment.

When the source VM sends an ARP/NS request to find out the IP to MAC mapping of a remote VM, if the local SARP proxy doesn't have the ARP cache for the target IP address or the cache entry has expired, the



ARP/NS request is propagated to all Access Segments which might have VMs in the same virtual network as the originating VM, including the east Access Segment.

The destination VM responds to the ARP/NS request and transmits an ARP reply (IPv4) or Neighbor Advertisement (IPv6) having the IP-D to MAC-D mapping.

The east SARP proxy functions as the proxy ARP of its Local VMs. The east SARP proxy modifies the ARP reply or NA message's source MAC-D to MAC-E and forwards the modified ARP reply or NA message to all the SARP proxies.

The West SARP Proxy forwards the modified ARP reply message to the source VM.

The west SARP proxy can also function as an IP<->MAC cache of the Remote VMs. By doing so, it significantly reduces the volume of the ARP/ND transmission over the network.

When the west SARP proxy caches the IP<-> MAC mapping entries for remote VMs, the timers for the entries to expire should be set relatively small to prevent stale entries due to remote VMs being moved or deleted. For environment where VMs move more frequently, it is not recommended for SARP Proxy to cache the IP<-> MAC mapping entries of remote VMs.

### 3.1.3. Gratuitous ARP and Unsolicited Neighbor Advertisement (UNA)

Hosts (or VMs) send out Gratuitous ARP (IPv4) and Unsolicited Neighbor Advertisement - UNA (IPv6) for other nodes to refresh IP<->MAC entries in their cache.

The local SARP processes the Gratuitous ARP or UNA in the same way as the ARP reply or IPv6 NA, i.e. replace the source MAC with its own MAC.

## 3.2. Data Plane: Packet Transmission

### 3.2.1. Local Packet Transmission

When a VM transmits packets to a destination VM that is located at the same site, there is no change in the data plane. The packets are sent from (IP-S, MAC-S) to (IP-D, MAC-D).

### 3.2.2. Packet Transmission Between Sites

Packets that are sent between sites traverse the SARP proxy of both sites. In our example, all packets sent from the VM located at the west site to the destination VM located at the east site traverse the west SARP proxy and the east SARP proxy.

The source VM follows its ARP table and sends packets to (IP-D, MAC-E) destination addresses and with (IP-s, MAC-S) as the source addresses.

The west SARP proxy can either 1) simply forward the data frame to MAC-E, or 2) replace the packet source address to its own source address (MAC-W), keeps the destination address to be (MAC-E), and forwards the packet to the east proxy SARP.

It is recommended for west SARP proxy to replace Source Address with its own if the "interconnecting segment" has address learning enabled. Otherwise nodes in the "interconnecting segment" can't learn the address of the switch on which west SARP proxy is running unless the switch sends out frames periodically.

When the east proxy SARP receives the packet, it replaces the destination MAC address to be (MAC-D) based on the packet destination IP (i.e., IP-D), but it does not change the source MAC addresses. When the destination VM receives the packet, the Source Address field would be the MAC address of the VM on the west side or the MAC address of the west side SARP proxy,

Noted: it is common for data center network to have security policies to enforce some VMs can communicate with each other, and some VMs can't. Most likely, those policies are configured by VM's IP addresses. Even though the originating VM's MAC address might be lost when the packet arrives at the destination VM, the originating VM's IP address is still present in the data packets for security policy to be enforced.

Noted: for the option which doesn't need west SARP to change source MAC of the data frames, the originating VM's MAC will be present when the data frames arrive at the destination VMs. Therefore, this option is valuable when hosts/VMs need to validate source VMs MAC addresses to comply any policies imposed.

Noted: Most hosts/VMs refresh its IP<->MAC mapping cache, with the Source MAC and Source IP of a received data frame. For the option which west SARP changes data frame's source MAC with its own MAC address, the destination VM's IP<->MAC cache can be refreshed with

the valid mapping of the Source-VM-IP <->West-SARP-MAC. For the option of West SARP not changing source MAC, the destination VM has to turn off the learning of IP<->MAC mapping from the received data frames.

### 3.3. VM Migration

#### 3.3.1. VM Local Migration

When a VM migrates locally within its Access segment, the SARP protocol is not required to perform any action. VM migration is resolved entirely by the Layer 2 mechanisms.

#### 3.3.2. VM Migration from One Site to Another

In our example, the VM migrates from the west site to the east site while maintaining its MAC and IP addresses.

VM migration might affect networking elements based on their respective location:

- Origin site (west site)
- Destination site (east site)
- Other sites

Origin site:

The Origin site is the site where the VM is before migration. It is the west site in our example.

Before the VM (IP=IP-D, MAC=MAC-D) is moved, all VMs at the west site that have an ARP entry of IP-D in their ARP table have the (IP-D to MAC-D) mapping. VMs on any other "Access Segments" will have ARP entry of (IP-D to MAC-W) mapping where MAC-W is the MAC address of the SARP proxy on the West Access Segment.

After the VM (IP-D) in the West Site moves to East Site, if there is gratuitous ARP (IPv4) or Unsolicited Neighbor Advertisement (IPv6) sent out by the destination hypervisor for the VM (IP-D), then the IP<->MAC mapping cache of VMs on all Access Segments will be updated by (IP-D to MAC-E) where MAC-E is the MAC address of the SARP proxy on the East Site. If there isn't any gratuitous ARP or Unsolicited Neighbor Advertisement sent out by the destination hypervisor, the

IP<->MAC cache on the VMs in west site (and other sites) will eventually aged out.

Until IP<->MAC mapping cache tables are updated, the source VMs from the west site continue sending packets to MAC-D. Switches at the west site are still configured with the old location of MAC-D. This can be resolved by VM manager sending out a fake gratuitous ARP or Unsolicited Neighbor Advertisement on behalf of destination Hypervisor, shorter aging timer configured for IP<->MAC cache table, or by redirecting the packets to the proxy SARP of the west site.

#### Destination Site:

The destination site is the site to which the VM migrated, the east site in our example.

Before any gratuitous ARP or Unsolicited Neighbor Advertisement messages are sent out by the destination hypervisor, all VMs at the east site (and all other sites) might have (IP-D to MAC-W) mapping in their IP<->MAC mapping cache. IP<->MAC mapping cache is updated by aging or by a gratuitous ARP or UNA message sent by the destination hypervisor. Until IP<->MAC mapping caches are updated, the source VMs from the east site continue to send packets to MAC-W. This can be resolved by VM manager sending out a fake gratuitous ARP/UNA immediately after the VM migration, or redirecting the packets from the SARP proxy of the east site to the migrated VM by updating the destination MAC of the packets to MAC-D.

#### Other Sites:

All VMs at the other sites that have an ARP entry of IP-D in their ARP table have the (IP-D to MAC-W) mapping. ARP mapping is updated by aging or by a gratuitous ARP message sent by the destination hypervisor of the migrated VM and modified by the SARP proxy of the east site (IP-D to MAC-E) mapping. Until ARP tables are updated, the source VMs from the west site continue sending packets to MAC-W. This can be resolved by redirecting the packets from the SARP proxy of the west site to the SARP proxy of the east site by updating the destination MAC of the packets to MAC-E.

#### 3.3.2.1. Impact to IP<->MAC Mapping Cache Table of VMs being moved

When a VM (IP-D) is moved from one site to another site, its IP<->MAC mapping entries for VMs located at the other sites (i.e. neither east site nor west site) are still valid, even though most Guest OSs (or VMs) will refresh their IP<->MAC cache after migration.

The VM (IP-D)'s IP<->MAC mapping entries for VMs located at east site, if not refreshed after migration, can be kept with no change until the ARP aging time since they are mapped to MAC-E. All traffic originated from the VM (IP-D) in its new location to VMs located at the east site traverses the SARP proxy of the east Site. The ARP/UNA sent by the SARP proxy of the east site or by the VMs on east side can always refresh the corresponding entries in the VM (IP-D)'s IP<->MAC cache .

The VM (IP-D)'s ARP entries (i.e. IP to MAC mapping) for VMs located at west sites will not be changed either until the ARP entries age out or new data frames are received from the remote sites. Since all MAC addresses of the VMs located at the west site are unknown at the east site. All unknown traffic from the VM is intercepted by the SARP proxy of the east site and forwarded to the SARP proxy of the west site (just for ARP aging time). This can be resolved by the east SARP proxy having mapping entries for VMs in the west side. Upon receiving unknown packets, it can update the migrating VM with the new IP to MAC mapping by sending a modified gratuitous ARP with (IP-D to MAC-W) mapping.

Note that overlay networks providing the Layer 2 network virtualization services configure their Edge Device MAC aging timers to be greater than the ARP request interval.

### 3.4. Multicast and Broadcast

To be added in a future version of this document

### 3.5. Non IP packet

To be added in a future version of this document

### 3.6. IP<->MAC caching on SARP Proxy

ARP/NS Requests for a VM located at a remote site require flooding messages over the interconnecting network to all sites which have enabled the virtual network on which the VM belongs to. This scenario is described in details at 3.1.2. In such cases, SARP caching can reduce the number of ARP/ND transmissions over interconnecting networks.

In the example presented at section 3.1.2. the source VM is located at the west site and the destination VM is located at the east site. When the source VM sends an ARP or Neighbor Solicitation request to discover the IP to MAC mapping of the remote VM, the request can be intercepted by the west SARP proxy.

The west SARP proxy learns or refreshes the source IP to source MAC mapping and looks up the IP to MAC translation of the destination IP. If the destination IP entry is found and is valid, the west SARP proxy replies with an ARP reply or Neighbor Advertisement without propagating the packet to other sites. Otherwise, the packet is propagated to all sites which have the virtual network enabled including the east site.

The propagated ARP/NS request is intercepted again by the east SARP proxy. It learns or refreshes the source IP to source MAC mapping and looks up the destination IP to MAC translation. If the destination IP entry is found and is valid the SARP proxy replies with an ARP reply or NA without propagating the ARP request to the east site. Otherwise, the ARP/NS request is broadcasted within the east site.

The destination VM responds to the ARP/NS request and transmits an ARP reply or NA having the IP-D to MAC-D mapping.

The east side SARP proxy intercepts the ARP reply or NA and learns or refreshes the Destination IP to Destination MAC mapping, replace the source MAC with its own MAC before sending the ARP reply or NA to the west SARP proxy (so that requesting VM can learn the IP-D to MAC-E mapping).

The West SARP Proxy intercepts the ARP reply or NA and learns or refreshes the Destination IP to Destination MAC mapping and propagates the ARP reply to the source VM.

The SARP proxies maintain an ARP caching table of IP to MAC mapping for all recent ARP/NS requests and replies. This table allows the SARP proxy to respond with low latency to the ARP/NS requests sent locally and avoid the broadcast transmissions of such requests over the transport network and all over the broadcast domains at the remote sites.

### 3.7. High availability and load balancing

The SARP proxy is located at the boundary where the local Layer 2 infrastructure connects to the interconnecting network. All traffic from the local site to the remote sites traverses the SARP proxy. The

SARP proxy is subject to high availability and bandwidth requirements.

The SARP architecture supports multiple SARP proxies connecting a single site to the transport network. In SARP architecture all proxies can be active and can backup one another. The SARP architecture is robust and allows the network administrator to allocate proxies according to the bandwidth and high availability requirements.

Traffic is segregated between SARP proxies by using VLANs. An SARP proxy is the Master-SARP proxy of a set of VLANs and the Backup-SARP proxy of another set of VLANs.

For example the SARP proxies of the west site (as Figure 1 depicts) are SARP proxy-1 and SARP proxy-2. The west site supports VLAN-1 and VLAN-2 while SARP proxy-1 is the Master SARP proxy of VLAN-1 and the Backup proxy of VLAN-2 and SARP proxy-2 is the Master SARP proxy of VLAN-2 and the Backup SARP proxy of VLAN-1. Both proxies are members of VLAN-1 and VLAN-2.

The Master SARP proxy updates its Backup proxy with all the ARP reply messages. The Backup SARP proxy maintains a backup database to all the VLANs that it is the Backup SARP proxy.

The Master and the Backup SARP proxies maintain a keepalive mechanism. In case of a failure the Backup proxy becomes the Master SARP proxy. The failure decision is per VLAN. When the Master and the Backup proxies switchover, the backup SARP proxy can use the MAC address of the Master SARP proxy. The backup SARP proxy sends locally a gratuitous ARP message with the MAC address of the Master SARP proxy to update the forwarding tables on the local switches. The backup SARP proxy also updates the remote SARP proxies on the change.

### 3.8. SARP Interaction with Overlay networks

SARP interaction with overlay networks providing L2 network virtualization (such as IP, VPLS, Trill, OTV, NVGRE and VxLAN) is efficient and scalable.

The mapping of SARP to overlay networks is straightforward. The VM does the destination IP to SARP proxy MAC mapping. The mapping of the proxy MAC to its correct tunnel is done by the overlay networks. SARP significantly scales down the complexity of the overlay networks and transport networks by reducing the mapping tables to the number of SARP proxies.

#### 4. Conclusions

SARP distributes the Layer 2 Forwarding Information Base (FIB) from the edge devices (functioning as SARP proxies) to the VMs. By doing so, it significantly reduces table sizes on the edge devices. The source VM maintains the mapping of its destination VMs to the destination site/cloud in the ARP table. The destination VM IP is translated to the destination MAC address of the SARP proxy at the destination site. The SARP proxies only maintain Layer 2 FIB of local VMs and remote edge devices.

SARP proxies can support FAST VM migration and provide minimum transition phase. When SARP proxy indicates or is informed of VM migration, it can update all its peers and trigger a fast update.

SARP seamlessly supports Layer 2 network virtualization services over the overlay network and significantly reduces their complexity in terms of table size and performance. The overlay networks are only required to map MAC addresses of the SARP proxies to the correct tunnel.



## 5. Security Considerations

The SARP proxies are located at the boundaries where the local Layer 2 infrastructure connects to its Layer 2 cloud. The SARP proxies interoperate with overlay network protocols that extend the Layer-2 subnet across data centers or between different systems within a datacenter.

SARP control plane and data plane are traversed by the overlay network hence SARP does not expose the network to additional security threats.

SARP proxies may be exposed to Denial of Service (DoS) attacks by means of ARP/ND message flooding. Thus, the SARP proxies must have sufficient resources to support the SARP control plane without making the network more vulnerable to DoS than without SARP proxies.

SARP adds security to the data plane by hiding all the local layer 2 MAC addresses from potential attacker located at the remote clouds. The only MAC addresses that are exposed at remote sites are the MAC addresses of the SARP proxies.

## 6. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

## 7. References

### 7.1. Normative References

- [ARP] Plummer, D., "An Ethernet Address Resolution Protocol", RFC 826, November 1982.
- [ND] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

### 7.2. Informative References

- [ARMD] Narten, T., Karir, M., Foo, I., " Problem Statement for ARMD", draft-ietf-armd-problem-statement, February 2012.

## 8. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

### Authors' Addresses

Youval Nachum  
Marvell  
6 Hamada St.  
Yokneam, 20692 Israel  
Email: youvaln@marvell.com

Linda Dunbar  
Huawei Technologies  
5430 Legacy Drive, Suite #175  
Plano, TX 75024, USA  
Phone: (469) 277 5840  
Email: ldunbar@huawei.com

Ilan Yerushalmi  
Marvell  
6 Hamada St.  
Yokneam, 20692 Israel  
Email: yilan@marvell.com

Tal Mizrahi  
Marvell  
6 Hamada St.  
Yokneam, 20692 Israel  
Email: talmi@marvell.com



Network Working Group  
INTERNET-DRAFT  
Updates RFC 2845 (if approved)  
Intended Status: Standards Track

H. Rafiee  
Hasso Plattner Institute  
M. v. Loewis  
Hasso Plattner Institute  
C. Meinel  
Hasso Plattner Institute  
September 30, 2012

Expires: March 2013

Transaction SIGNature (TSIG) using CGA Algorithm in IPv6  
<draft-rafiiee-cga-tsig-00.txt>

#### Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 30, 2013.

#### Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved. This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as

described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Abstract

The first step of Transaction SIGNature (TSIG) (RFC 2845) is to generate a shared secret and exchange it manually between a DNS server and a host. This document, CGA-TSIG, proposes a possible way to automate the now manual process for the authentication of a node with a DNS server during the DNS Update process by using the same parameters as are used in generating a secure address in IPv6 networks, i.e., Cryptographically Generated Addresses (CGA) (RFC 3972). CGA-TSIG facilitates this authentication process and reduces the time needed for DNS Updates. The current signature generation process and verification mechanism in TSIG are thus replaced with CGA. This algorithm is added, as an extension, to TSIG to eliminate the human intervention needed for generation and exchange of keys between a DNS server and a host when SECure Neighbor Discovery (SEND) (RFC 3971) is used.

## Table of Contents

1	Introduction . . . . .	3
2	Conventions used in this document . . . . .	3
3	Algorithm Overview . . . . .	3
3.1	CGA Generation Algorithm . . . . .	4
3.2.	Modification to TSIG protocol . . . . .	4
3.2.1.	Modified TSIG Record format . . . . .	5
3.2.1.1	Generation of the DNS Update request/response . . .	6
3.2.1.2	Verification of the DNS Update request/response . .	7
4	Security Considerations . . . . .	10
4.1	IP Spoofing and Reflector Attacks . . . . .	10
4.2	DNS Dynamic Update Spoofing . . . . .	10
4.3	Resolver Configuration Attack . . . . .	10
4.4	Shared Secret (key pairs) Exposing . . . . .	10
4.5	Replay attack . . . . .	10
5	IANA Considerations . . . . .	10
6	Conclusions . . . . .	11
7	References . . . . .	11
7.1	Normative References . . . . .	11
7.2	Informative References . . . . .	12
8	Acknowledgments . . . . .	12
	Appendix A. . . . .	12
	A.1. Copyright . . . . .	12
	Authors' Addresses . . . . .	13

## 1 Introduction

Transaction SIGNature (TSIG) [RFC2845] is a protocol that provides endpoint authentication and data integrity by using one-way hashing and shared secret keys to establish a trust relationship between two hosts which, can be, either a client and a server or two servers. The TSIG keys are manually exchanged between these two hosts and they must be maintained in a secure manner. This protocol is used to secure a Dynamic Update or to give assurance to the slave named server that the zone transfer is from the original master named server and that it has not been spoofed by hackers. It does this by verifying the signature with a cryptographic key shared with that of the receiver. The TSIG protocol can be extended using newly defined algorithms. This document defines an algorithm based on the Cryptographically Generated Addresses (CGA) [RFC3972]. CGA is one of the important options in SEcure Neighbor Discovery (SEND) [RFC3971] that can easily provide nodes with the necessary proof of address ownership by providing a cryptographic binding between a host and its IP address without the introduction of any new infrastructure.

## 2 Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. 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 characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

## 3 Algorithm Overview

CGA is a one-way hashing algorithm used to generate Interface IDs for IPv6 addresses in a secure manner. An interface ID consists of the rightmost 64 bits of the 128 bit IPv6 address. CGA verifies the address ownership of the sender by finding a relationship between the sender's IP address and his public key [1,2].



Figure 1 IPv6 addresses

### 3.1 CGA Generation Algorithm

A node proceeds with the following steps to generate the CGA:

1. Key pairs, called public/private keys, and a random number, called a modifier, are generated [key pair format: Section 3. RFC3972] It is recommended that key pairs be generated on the fly for the following reasons:
  - To decrease the randomization of IP addresses
  - To eliminate the need to have the keys manually generated and saved in a particular path, in the node, before the start of IP address generation
  - To decrease the chance of private key theft
2. The modifier is concatenated with other parameters such as a zero value prefix (64 bits), a zero value collision count (1 bit) and the RSA public key

+-----+-----+-----+-----+			
Modifier	Subnet Prefix	collision count	Public key
(2 octets)	(8 octets)	(1 bit)	(variable)
+-----+-----+-----+-----+			

Figure 2 CGA Parameters

3. The Secure Hash Algorithm (SHA1) is executed using the output from step 2. The first leftmost 112 bits of the resulting digest is called Hash2.

4. The computational complexity of Hash2 depends on the Sec value. The Sec is an unsigned 3-bit integer having a value between 0 and 7 (0 being the least secure while 7 the most) which indicates the security level of the generated address against brute-force attacks. The 16xSec leftmost bits of Hash2 are compared to zero. If the condition is not met, the modifier is incremented by one and steps 2 through 4 are repeated. If the condition is met, the next step is executed

5. The modifier is concatenated with the prefix, the collision count, and the public key. SHA1 is executed using that output to create Hash1. The CGA algorithm then uses the leftmost 64 bits from Hash1 and sets the first leftmost 3 bits to the sec value. It also sets bits 7 and 8 (bits u and g) and calls this the Interface ID (IID)

6. The subnet prefix is then concatenated with the IID and the Duplicate Address Detection (DAD) process is executed in order to detect address collision on the network. The node then includes the CGA parameters (modifier, subnet prefix, collision count, public key) with the messages to give other nodes the ability to verify the address ownership of the sender by finding a relationship between the sender's IP address and his public key.

### 3.2. Modification to TSIG protocol

Normally, to initiate a secure DNS Update process between a DNS server and a host (another DNS server or a client), a minimum of four messages are required to establish a secure channel. A modification to RFC-2845, CGA-TSIG, decreases the number of messages needed in the exchange. The messages used in RFC-2930 (TKEY RR) are not needed when CGA-TSIG is used.

The CGA-TSIG extension uses the creation of a TSIG Resource Record (RR). This RR uses the same data as used to generate a new IP address in a node-- for example, the key pairs (public/private keys), and the output value of the CGA generation function (Interface ID). It is recommended that these values be cached in the node's memory for later use.

### 3.2.1. Modified TSIG Record format

The modified TSIG RR uses the same format as other RRs in use in the DNS field. This is explained in section 3.2.1 RFC-1035, where the algorithm type must be set to TSIG. The RDATA is also extended in order to store the CGA parameters and the modified CGA signature. The RDATA's algorithm type must be set to CGA-TSIG, a detailed explanation of the RDATA standard fields can be found in section 2.3 RFC-2845. This document focuses only on the new extensions added to RDATA. These new fields are CGA-TSIG Len and CGA-TSIG DATA.

Algorithm type (CGA-TSIG)
Time Signed
Fudge
MAC Size
Mac
Original ID
Error
OTHER LEN
OTHER DATA
CGA-TSIG Len
CGA-TSIG DATA

Figure 3 Modified TSIG RDATA



CGA-TSIG DATA Field Name	Data Type	Notes
Algorithm type	u_int16_t	Name of the algorithm

Parameters Len	Octet	[RFC3972] RSA (by default) CGA
Parameters	variable	the length of CGA parameters CGA
Signature Len	Octet	Section 3.1 this document CGA
CGA Signature	variable	the length of CGA signature
		Section 3.2.1 This document

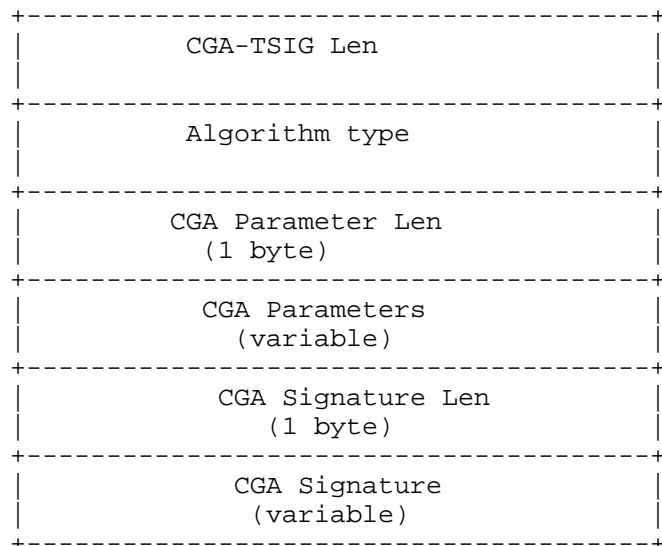


Figure 4 CGA-TSIG Len and CGA-TSIG DATA

#### 3.2.1.1 Generation of the DNS Update request/response

Both the DNS update request and response messages must contain the CGA-TSIG option. To generate the CGA-TSIG DATA, a DNS server and a host must follow steps 1 and 2. In the case where key pairs and CGA parameters are cached during the generation of the IP address by using SEND, the value for the first two steps can be obtained from cache.

1. Execute steps 1 through 4 of section 3.1. It is recommended that the same algorithm be used to generate both CGA key pairs and to generate and sign the CGA in the CGA-TSIG DATA Field. In the case of multiple DNS servers (authentication of two DNS servers), there are three possible scenarios with regard to the authentication process, which differs from that of the authentication of a node (client) with one DNS server because of the need for human intervention. a. Add DNS servers' IP address to a slave configuration file A DNS server administrator should only manually add the IP address of the master DNS server to the configuration file of the slave DNS server. When the DNS update message is processed, the slave DNS server can authenticate the master DNS server based on the source IP address and then prove the ownership of this address by using CGA. This scenario is valid until the IP address in any of these DNS servers changes. To automate this step's process, the DNS Update message sender's public key may be saved on the other DNS server after the source IP address has been successfully verified for the first time. In this case, when the sender generates a new IP address by executing the CGA

algorithm using the same public key, the other DNS server can still verify it and add its new IP address to the DNS configuration file automatically.

b. Manually exchange the public/private keys An administrator of the DNS servers may need to manually save the public/private keys of a master DNS server in the slave DNS server. This approach does not have the disadvantage of the first approach since, any time any DNS server wants to change its IP address, it will use the public/private keys.

c. Retrieve public/private keys from a third party Trusted Authority (TA) The message exchange option of SEND [RFC3971] may be used for the retrieval of the third party certificate. This may be done automatically from the TA by using the Certificate Path Solicitation and the Certificate Path Advertisement messages. Like in scenario b, saving the certificate on the DNS server for later use in the generation of its address or in the DNS update process. In this case, whenever any of these servers wants to generate a new IP address, the DNS update process can still be done automatically without the need for human intervention.

2. Generate signature For signature generation, all CGA parameters (modifier, public key, collision count and subnet prefix), that are concatenated with the DNS update message and the Time Signed field, are signed by using a RSA algorithm and the private key which was generated in the first step. This signature must be added as an extended option to the TSIG RDATA field. Time Signed is the same timestamp as is used in RDATA. This value is the UTC date and time value obtained from the signature generator. This approach will prevent replay attacks by changing the content of the signature each time a node wants to send a DNS Update Request. The Update Message contains all of the DNS update message with the exclusion of the TSIG Resource Records (RRs). A DNS update message consists of a header, a zone, a prerequisite, an update and additional data. The header contains the control information [RFC2136], the zone identifies the zones to which this update should be applied [Section 4.1.2 RFC1035], the prerequisite prescribes the RRs that must be in the DNS database, the update contains the RR that needs to be modified or added and the additional data is the data that is not part of the DNS update, but is necessary in order to process this update.

Modifier (16 bits)	Subnet Prefix (64 bits)	collision count (1 bit)	Public key (variable)
Time Signed	DNS Update Message		

Figure 5 CGA-TSIG Signature

### 3.2.1.2 Verification of the DNS Update request/response Sender authentication is necessary to prevent attackers from making

unauthorized modifications to DNS servers by use of spoofed DNS Update messages.

1. Check the subnet prefix The leftmost 64 bits of IPv6 addresses constitute the subnet prefix. The receiver obtains the subnet prefix from the source IP address in the sender's message. Then, the subnet prefix is obtained from the CGA parameters in the TSIG RDATA field of the received message. A comparison is then made between these two subnet prefixes. If the subnet prefixes match step 2 is executed, otherwise the node is considered as an attacker and the message should be discarded without further action.

2. Check the Time Signed The Time Signed value is obtained from the TSIG RDATA and is denoted  $t_1$ . The current system time is then obtained and converted to UTC time and is denoted  $t_2$ . If  $t_1$  is in the range of  $t_2$  and  $t_2$  minus 2 seconds (see formula 1, 2 seconds may vary according to the transmission lag time) step 3 is executed, otherwise, the message is considered a spoofed message and the message should be discarded without further action. The range of two seconds is used because the update message may experience a delay during its transmission over TCP or UDP. Both times must use UTC time to avoid any differences in the time based on different geographical locations.  $t_2 - 2 \leq t_1 \leq t_2$  (1)

3. Compare Hash1 to the Interface ID The receiver should obtain all CGA parameters from the TSIG RDATA field and execute SHA1 against them. The leftmost 64 bits of the resulting output constitutes Hash1. Hash1 is then compared to the rightmost 64 bits of the sender's IP address, which is known as the Interface ID (IID). Any differences in the first three leftmost bits of the IID (Sec value) and the u and the g bits (Section 3.1) are ignored. u and g are bits 7 and 8 of the first byte of the IID. If they match step 4 is executed, otherwise, the source is considered as a spoofed source IP address and the message should be discarded without further action.

4. Evaluate Hash2 with CGA parameters The receiver obtains the CGA parameters. The collision count and the subnet prefix are set to zero and SHA1 is executed on the resulting data in order to obtain a result of which the leftmost 112 bits are denoted as Hash2. The leftmost 16xSec bits of Hash2 are compared to zero. If the condition is met step 5 is executed, otherwise, the CGA parameters should be considered as spoofed CGA parameters and the message should be discarded without further action.

5. Verify the signature The signature contained in the TSIG RDATA field of the DNS update message should be verified. This can be done by retrieving the public key from the TSIG RDATA and using it to verify the signature. If the verification process is successful and the node does not want to update another node's RR, then the Update Message will be processed. If the signature verification is successful and the node wants to update another node's RRs, then step 6 is executed. If the verification fails, then the message should be

discarded without further action.

6. Verify the Source IP address If a node wants to update a/many RR(s) on another DNS server, like a master DNS server wanting to update RRs on the slave DNS server, the requester's source IP address must be checked against the one in the DNS configuration file. If it is the same the Update Message should be processed, otherwise, step 7 is executed.

7. Verify the public key The DNS server checks whether or not the public key retrieved from the TSIG RDATA is the same as what was saved manually by the administrator. If it is the same, step 8 is executed, otherwise, the message should be discarded without further action.

8. Re-generate the signature The DNS server retrieves the CGA parameters, the Time Signed and the Update Message from the content of the DNS Update Request. These fields are then concatenated. The algorithm type is obtained from the Other Data field of the TSIG RDATA which, by default, should be the RSA, and then a signature is generated using the private key of the sender obtained from the local storage in the DNS server (the key pairs manually saved by administrator). This signature is compared with the signature obtained from the TSIG RDATA. If there is a match, then the Update Message is processed, otherwise, the message should be discarded without further action.

## 4 Security Considerations

There are several attacks that CGA-TSIG can prevent. Here we evaluate some of these attacks.

### 4.1 IP Spoofing and Reflector Attacks

During the DNS Update process it is important for both communicating parties to know that the one they are communicating with is the owner of that IP address and that messages have not been sent from a spoofed IP address. This can be fulfilled by using the CGA algorithm that utilizes the node to verify the address ownership of the other node. The reflector attack is also a kind of distributed Denial of Service attack. It uses the IP address of the victim as a source of the DNS message and sends several queries to the DNS server which then redirects traffic to this victim thus keeping the victim busy processing these packets. Using the CGA signature and authentication approach will prevent this type of attack.

4.2 DNS Dynamic Update Spoofing Because the signature contains both CGA parameters and the DNS update message, proof is offered of the sender's address ownership (CGA parameters) and the validity of the update message.

4.3 Resolver Configuration Attack In CGA-TSIG, the DNS server or the client might not need further configuration. This may reduce the possibility for human errors being inserted into the DNS configuration file. Since this type of attack is predicated on human error, the chances of it occurring when our proposed extension is used are minimized.

4.4 Shared Secret (key pairs) Exposing On-the-fly key pair generation is recommended to decrease the chances of giving attackers unauthorized access to private keys on a node.

4.5 Replay attack Using Time Signed in the signature modifies the content of the signature each time the node generates it and sends it to DNS server. This value is the node's current time in UTC. If the attacker tries to spoof this value with another timestamp to show that the update message is current, the DNS server checks this message by verifying and regenerating the signature (when the private key of the other DNS server is manually set in this DNS server). In this case steps 2 and 8 of verification process fail. Therefore, this type of attack is also prevented.

## 5 IANA Considerations

The IANA has allowed for choosing new algorithm(s) for use in the TSIG Algorithm name. Algorithm name refers to the algorithm described in this document. The requirement to have this name registered with IANA is specified

## 6 Conclusions

In TSIG, not all processing is done automatically and some steps might need to be done offline. To address this issue, and to automate this process when Secure Neighbor Discovery (SEND) (RFC3971) is used, this document is introduced as an extension to the TSIG protocol (CGA-TSIG) in order to take advantage of the use of CGA for the DNS Update authentication process of a node within a DNS server. CGA-TSIG also decreases the number of messages needed in the exchange between the DNS server and the DNS client during the update process. This enhances the performance of the DNS update process. Since CGA does not need Public Key Infrastructure (PKI) framework to verify the node's address ownerships, the authentication of a node with a DNS server in the DNS update process is automated. This document also makes use of SEND for the authentication of two DNS servers against each other when processing DNS Update messages. However, the first step should be done manually the first time it is used to afford greater security for this process

## 7 References

### 7.1 Normative References

- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)," RFC 3972, March 2005.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.
- [RFC1035] Mockapetris, P., "Domain Names - Implementation And Specification", RFC 1035, November 1987.
- [RFC2136] Vixie, P. (Editor), Thomson, S., Rekhter, Y., Bound, J., "Dynamic Updates in the Domain Name System (DNS UPDATE)",

RFC 2136, April 1997.

## 7.2 Informative References

- [1] Aura, T., "Cryptographically Generated Addresses (CGA)", Lecture Notes in Computer Science, Springer, vol. 2851/2003, pp. 29-43, 2003.
- [2] Montenegro, G. and Castelluccia, C., "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," ISOC Symposium on Network and Distributed System Security (NDSS 2002), the Internet Society, 2002.

## 8 Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

## Appendix A.

### A.1. Copyright

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved. Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).



Authors' Addresses

Hosnieh Rafiee  
Hasso-Plattner-Institute  
Prof.-Dr.-Helmert-Str. 2-3  
Potsdam, Germany  
Phone: +49 (0)331-5509-546  
Email: rafiee@hpi.uni-potsdam.de

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

Dr. Martin von Loewis  
Hasso-Plattner-Institute  
Prof.-Dr.-Helmert-Str. 2-3  
Potsdam, Germany  
Email: martin.vonloewis@hpi.uni-potsdam.de

Rafiee, von Loewis & Meinel Expires March 2013

[Page 13]