Intarea Working Group
Internet-Draft
Intended status: Informational
Expires: February 13, 2015

R. Bonica
Juniper Networks
C. Pignataro
Cisco Systems
J. Touch
USC/ISI
August 12, 2014

A Widely-Deployed Solution To The Generic Routing Encapsulation (GRE) Fragmentation Problem
draft-bonica-intarea-gre-mtu-06

Abstract

This memo describes how many vendors have solved the Generic Routing Encapsulation (GRE) fragmentation problem. The solution described herein is configurable. It is widely deployed on the Internet in its default configuration.

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 February 13, 2015.

Copyright Notice

Table of Contents

1. Introduction

Generic Routing Encapsulation (GRE) [RFC2784] [RFC2890] can be used to carry any network layer protocol over any network layer protocol. GRE has been implemented by many vendors and is widely deployed in the Internet.

The GRE specification does not describe fragmentation procedures. Lacking guidance from the specification, vendors have developed implementation-specific fragmentation solutions. A GRE tunnel will operate correctly only if its ingress and egress nodes support compatible fragmentation solutions. [RFC4459] describes several fragmentation solutions and evaluates their relative merits.

This memo reviews the fragmentation solutions presented in [RFC4459]. It also describes how many vendors have solved the GRE fragmentation problem. The solution described herein is configurable, and has been widely deployed in its default configuration.

This memo addresses point-to-point unicast GRE tunnels that carry IPv4, IPv6 or MPLS payloads. All other tunnel types are beyond the scope of this document.

1.1. Terminology

The following terms are specific to GRE and are taken from [RFC2784]:

- GRE delivery header - an IPv4 or IPv6 header whose source address represents the GRE ingress node and whose destination address represents the GRE egress node. The GRE delivery header encapsulates a GRE header.

- GRE header - the GRE protocol header. The GRE header is encapsulated in the GRE delivery header and encapsulates GRE payload.

- GRE payload - a network layer packet that is encapsulated by the GRE header. The GRE payload can be IPv4, IPv6 or MPLS. Procedures for encapsulating IPv4 and IPv6 in GRE are described in [RFC2784] and [RFC2890]. Procedures for encapsulating MPLS in GRE are described in [RFC4023]. While other protocols may be delivered over GRE, they are beyond the scope of this document.

- GRE delivery packet - A packet containing a GRE delivery header, a GRE header, and GRE payload.

- GRE payload header - the IPv4, IPv6 or MPLS header of the GRE payload

- GRE overhead - the combined size of the GRE delivery header and the GRE header, measured in octets

The following terms are specific MTU discovery:

- link MTU (LMTU) - the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed over a link. LMTU is a unidirectional metric. A bidirectional link may be characterized by one LMTU in the forward direction and another MTU in the reverse direction.

- path MTU (PMTU) - the minimum LMTU of all the links in a path between a source node and a destination node. If the source and destination node are connected through an equal cost multipath (ECMP), the PMTU is equal to the minimum LMTU of all links contributing to the multipath.

- GRE MTU (GMTU) - the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed over a GRE tunnel without fragmentation of any kind. The GMTU is equal to the PMTU associated with the path between the GRE ingress and the GRE egress, minus the GRE overhead

- Path MTU Discovery (PMTUD) - A procedure for dynamically discovering the PMTU between two nodes on the Internet. PMTUD procedures for IPv4 are defined in [RFC1191]. PMTUD procedures for IPv6 are defined in [RFC1981].

The following terms are introduced by this memo:

- fragmentable packet - An IPv4 packet with DF-bit equal to 0 and whose payload is larger than 64 bytes

- ICMP Packet Too Big (PTB) message - an ICMPv4 [RFC0792] Destination Unreachable message with code equal to 4 (fragmentation needed and DF set) or an ICMPv6 [RFC4443] Packet Too Big message

2. Solutions

2.1. RFC 4459 Solutions

Section 3 of [RFC4459] identifies several tunnel fragmentation solutions. These solutions define procedures to be invoked when the tunnel ingress router receives a packet so large that it cannot be forwarded though the tunnel without fragmentation of any kind. When applied to GRE, these procedures are:

1. Discard the incoming packet and send an ICMP PTB message to the incoming packet’s source.

2. Fragment the incoming packet and encapsulate each fragment within a complete GRE header and GRE delivery header.

3. Encapsulate the incoming packet in a single GRE header and GRE delivery header. Perform source fragmentation on the resulting GRE delivery packet.

As per RFC 4459, Strategy 2) is applicable only when the incoming packet is fragmentable. Also as per RFC 4459, each strategy has its relative merits and costs.

2.2. A Widely-Deployed Solution

Many vendors have implemented a configurable GRE fragmentation solution. In its default configuration, the solution behaves as follows:

- When the GRE ingress node receives a fragmentable packet with length greater than the GMTU, it fragments the incoming packet and

encapsulates each fragment within a complete GRE header and GRE delivery header.

- When the GRE ingress node receives a non-fragmentable packet with length greater than the GMTU, it discards the packet and send an ICMP PTB message to the packet's source.

- When the GRE egress node receives a GRE delivery packet fragment, it silently discards the fragment, without attempting to reassemble the GRE delivery packet to which the fragment belongs.

In non-default configurations, the GRE ingress node can execute any of the procedures defined in RFC 4459.

The solution described above is widely-deployed on the Internet in its default configuration.

3. Implementation Details

This section describes how many vendors have implemented the solution described in Section 2.2.

3.1. General

The GRE ingress nodes satisfy all of the requirements stated in [RFC2784].

3.2. GRE MTU (GMTU) Estimation and Discovery

GRE ingress nodes support a configuration option that associates a GMTU with a GRE tunnel. By default, GMTU is equal to the MTU associated with next-hop toward the GRE egress node minus the GRE overhead.

Typically, GRE ingress nodes further refine their GMTU estimate by executing PMTUD procedures. However, if an implementation supports PMTUD for GRE tunnels, it also includes a configuration option that disables PMTUD. This configuration option is required to mitigate certain denial of service attacks (see Section 5).

The ingress node's GMTU estimate will not always reflect the actual GMTU. It is only an estimate. When a tunnel's GMTU changes, the tunnel ingress node will not discover that change immediately. Likewise, if the ingress node performs PMTUD procedures and tunnel interior nodes cannot deliver ICMP feedback to the tunnel ingress, GMTU estimates may be inaccurate.

3.3. GRE Ingress Node Procedures

This section defines procedures that GRE ingress nodes execute when they receive a packet whose size is greater than the relevant GMTU.

3.3.1. Procedures Affecting the GRE Payload

3.3.1.1. IPv4 Payloads

By default, if the payload is fragmentable, the GRE ingress node fragments the incoming packet and encapsulates each fragment within a complete GRE header and GRE delivery header. Therefore, the GRE egress node receives several complete, non-fragmented delivery packets. Each delivery packet contains a fragment of the GRE payload. The GRE egress node forwards the payload fragments to their ultimate destination where they are reassembled.

Also by default, if the payload is not fragmentable, the GRE ingress node discards the packet and sends an ICMPv4 Destination Unreachable message to the packet's source. The ICMPv4 Destination Unreachable message code equals 4 (fragmentation needed and DF set). The ICMPv4 Destination Unreachable message also contains an Next-hop MTU (as specified by [RFC1191]) and the next-hop MTU is equal to the GMTU associated with the tunnel.

The GRE ingress node supports a non-default configuration option that invokes an alternative behavior. If that option is configured, the GRE ingress node fragments the delivery header. See Section 3.3.2 for details.

3.3.1.2. IPv6 Payloads

By default, the GRE ingress node discards the packet and send an ICMPv6 [RFC4443] Packet Too Big message to the payload source. The MTU specified in the Packet Too Big message is equal to the GMTU associated with the tunnel.

The GRE ingress node supports a non-default configuration option that invokes an alternative behavior. If that option is configured, the GRE ingress node fragments the delivery header.See Section 3.3.2 for details.

3.3.1.3. MPLS Payloads

By default, the GRE ingress node discards the packet. As it is impossible to reliably identify the payload source, the GRE ingress node does not attempt to send an ICMP PTB message to the payload source.

The GRE ingress node supports a non-default configuration option that invokes an alternative behavior. If that option is configured, the GRE ingress node fragments the delivery header. See Section 3.3.2.

3.3.2. Procedures Affecting The GRE Deliver Header

3.3.2.1. Tunneling GRE Over IPv4

By default, the GRE ingress node does not fragment delivery packets. However, the GRE ingress node includes a configuration option that allows delivery packet fragmentation.

By default, the GRE ingress node sets the DF-bit in the delivery header to 1 (Don’t Fragment). However, the GRE ingress node also supports a configuration option that invokes the following behavior:

- when the GRE payload is IPv6, the DF-bit on the delivery header is set to 0 (Fragments Allowed)
- when the GRE payload is IPv4, the DF-bit is copied from the payload header to the delivery header

When the DF-bit on an IPv4 delivery header is set to 0, the GRE delivery packet can be fragmented by any node between the GRE ingress and the GRE egress.

If the delivery packet is fragmented, it is reassembled by the GRE egress.

3.3.2.2. Tunneling GRE Over IPv6

By default, the GRE ingress node does not fragment delivery packets. However, the GRE ingress node includes a configuration option that allows this.

If the delivery packet is fragmented, it is reassembled by the GRE egress.

3.4. GRE Egress Node Procedures

By default, the GRE egress node silently discards GRE delivery packet fragments, without attempting to reassemble the GRE delivery packets to which the fragments belongs.

However, the GRE egress node supports a configuration option that allows it to reassemble GRE delivery packets.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

In the GRE fragmentation solution described above, either the GRE payload or the GRE delivery packet can be fragmented. If the GRE payload is fragmented, it is typically reassembled at its ultimate destination. If the GRE delivery packet is fragmented, it is typically reassembled at the GRE egress node.

The packet reassembly process is resource intensive and vulnerable to several denial of service attacks. In the simplest attack, the attacker sends fragmented packets more quickly than the victim can reassemble them. In a variation on that attack, the first fragment of each packet is missing, so that no packet can ever be reassembled.

Given that the packet reassembly process is resource intensive and vulnerable to denial of service attacks, operators should decide where reassembly process is best performed. Having made that decision, they should decide whether to fragment the GRE payload or GRE delivery packet, accordingly.

PMTU Discovery is vulnerable to two denial of service attacks (see Section 8 of [RFC1191] for details). Both attacks are based upon on a malicious party sending forged ICMPv4 Destination Unreachable or ICMPv6 Packet Too Big messages to a host. In the first attack, the forged message indicates an inordinately small PMTU. In the second attack, the forged message indicates an inordinately large MTU. In both cases, throughput is adversely affected. On order to mitigate such attacks, GRE implementations includes a configuration option to disable PMTU discovery on GRE tunnels. Also, they can include a configuration option that conditions the behavior of PMTUD to establish a minimum PMTU.

6. Acknowledgements

The authors would like to thank Fred Baker, Fred Detienne, Jagadish Grandhi, Jeff Haas, Vanitha Neelamegam, John Scudder, Mike Sullenberger and Wen Zhang for their constructive comments. The authors also express their gratitude to Vanessa Ameen, without whom this memo could not have been written.

7. References

7.1. Normative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, September 2000.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, March 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

7.2. Informative References

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, April 2006.

Authors' Addresses

Ron Bonica
Juniper Networks
2251 Corporate Park Drive Herndon
Herndon, Virginia 20170
USA

Email: rbonica@juniper.net

Carlos Pignataro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, North Carolina 27709
USA

Email: cpignata@cisco.com

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, California 90292-6695
USA

Phone: +1 (310) 448-9151
Email: touch@isi.edu
URI: http://www.isi.edu/touch

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: August 17, 2014

Z. Cao
Q. Fu
L. Deng
China Mobile

February 13, 2014

Data Plane Processing Acceleration Framework
draft-cao-dataplane-acceleration-framework-00

Abstract

It is getting popular to running data applications over general purpose hardware/chipsets, instead of customized and dedicated hardware/chipset. This way further decouples the software functions from the hardware. But moving data processing intensive applications to general purpose hardware is still challenging, although the industry has supplied some proprietary solutions. This document discusses the problems of data plane acceleration and proposes its framework.

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 August 17, 2014.

Copyright Notice

Table of Contents

1. Introduction

The need of running network data processing functions over general purpose hardware/chipset (e.g., X86, PPC, etc) is multi-folded.

1. Decoupling software functions from hardware. Traditional network devices are built upon dedicated or deep customized hardware and chipsets. This way restricts the flexibility of both service providers and network operators.

2. Network Function Virtualization (NFV). NFV is an initiative of ETSI to virtualize the network functions to the overlay on top of the virtualization layer. It provides network elasticity in that the network functions can be scaled up/down according to the traffic load. NFV solutions often bundle with the virtual switches to provide VM-VM communications. Theses virtual switches are running on top of the servers that bear the network functions. Therefore, the need to accelerate the data processing efficiency is indispensable.

3. Service Time-to-Market . Via the software and hardware decoupling, the speed to provide new services (TTM) is greatly enhanced. Since more and more services would like to have the most convenient time to market, they would also like to move data processing functions on top of general purpose hardware/chipsets.

4. Capex and Opex pressure. Having the network functions running over general purpose device will help operators to cut down their Capex and Opex.

5. Cost-performance targets: software development, debug and integration is simplified; processor resource utilization is improved because the control plane and data plane can be distributed among cores with greater flexibility; development schedule risk is minimized and software maintenance is much easier with a common code base and a single development environment.

2. Requirements Language

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].

3. DPA Framework

NF (Network Function): A functional building block within an operator’s network infrastructure, which has well-defined external interfaces and a well-defined functional behaviour. Note that the totality of all network functions constitutes the entire network and services infrastructure of an operator/service provider. In practical terms, a Network Function is today often a network node or physical appliance. [Quoted from ETSI NFV]

3.1. Framework

The framework is depicted in Figure 1. Framework.

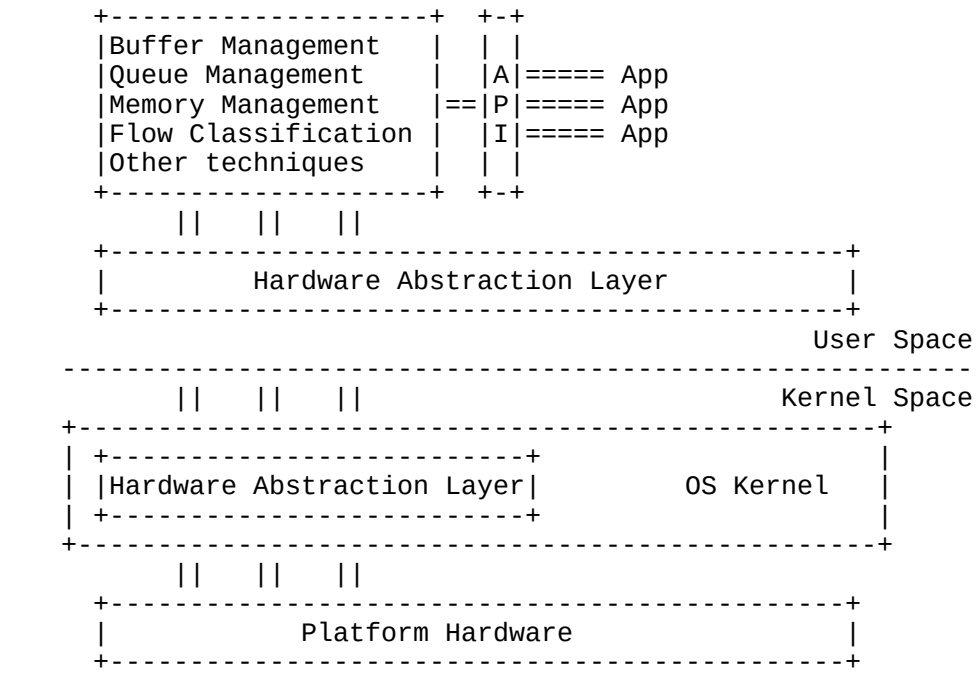


Figure 1

3.2. Components

The DPA may include the following components.

Memory/Buffer Manager. The Memory/Buffer Manager is responsible for allocating NUMA-aware pools of objects in memory and balancing memory bandwidth utilization across the channels. Such management can significantly reduces the amount of time the operating system must spend allocating and de-allocating buffers.

Queue Manager. The Queue Manager is responsible for queue scheduling. The ultimate goal of the Queue Manager is to allow different software components to process packets, while avoiding unnecessary wait times.

Flow Classification. The Flow Classification component is an efficient mechanism for generating a hash used to quickly combine packets into flows, which enables faster processing and greater throughput.

Poll Mode Drivers. The Poll Mode Drivers is capable of speeding up the packet pipeline for 1 GbE and 10 GbE ethernet controllers by

receiving and transmitting packets without the use of asynchronous, interrupt- based signaling mechanisms, which have a lot of overhead.

Environment Abstraction Layer. The Environment Abstraction Layer provides an abstraction to platform-specific initialization code, which eases application porting effort. The EAL provides access to low-level resources (hardware, memory space, logical cores, etc.) through a generic interface that hides the environment specifics from the applications and libraries.

3.3. Protocol Portfolio

On one hand, for the data plane, DPA should provide an efficient stack for common protocols utilized by various internet applications, including but not limited to:

1. Link layer: Layer 2 switch, VLAN.

2. Network layer: IPv4 and IPv6 for packet routing; MPLS and GRE/GTP for tunneled routing; IPsec, TLS/DTLS, NAT and QoS support for security and management features.

3. Transport layer: SCTP/MPTCP as well as TCP and UDP, for multi-homing/stream traffic.

4. Application layer: SSL termination for remote administration of virtualized device.

On the other hand, for the control plane, DPA should provide an efficient stack for common protocols utilized by various network devices/ISPs for improved operation and Management, including: NetFlow, sFlow, IPFIX, SPAN, RSPAN for VM traffic monitory, LACP, STP and openflow for L2/L3 management.

4. Existing Work - Intel DPDK

This section introduces DPDK [DPDK].

Intel Data Plane Development Kit (DPDK) is a set of libraries and drivers for fast packet processing on x86 platforms. It runs mostly in Linux userland.The idea of DPDK has significantly advanced the concept of consolidation of data and control planes on a general purpose processor. Such idea greatly boosts packet processing performance and throughput by providing Intel architecture-optimized libraries to accelerate L3 forwarding, yielding performance that scales linearly with the number of cores, in contrast to native Linux.

The Intel DPDK contains a growing number of libraries, whose source code is available for developers to use and/or modify in a production network element. Likewise, there are various usecase examples, such as L3 forwarding, load balancing, and timers, that help reduce development time. The libraries can be used to build applications based on "run-to completion" or "pipeline" models, enabling the equipment provider’s application to maintain complete control.

the Intel DPDK software is also available to aid in the development of I/O intensive applications running in a virtualized environment. This combination allows application developers to achieve near-native performance.

The Intel DPDK provides a simple framework for fast packet processing in data plane application. Developers may use the code to understand some of the techniques employed, to build upon for prototyping, or to add their own protocol stacks. SR-IOV features are also used for hadware-based I/O sharing in I/O virtualization (IOV) mode. Therefore, it is possible to partition intel 82599 10 Gb Ethernet controller NIC resources logically and expose them to a VM as a virtual function

Furthermore, 6WIND has developed a number of value-added enhancements to the Intel DPDK library that provide increased system functionality and performance compared to the baseline software. These value-added enhancements include the following aspects.

Hige-performance software crypto support, implemented via the Intel Advanced Encryption Standard New Instructions (Intel AES-NI) in the Intel Xeon processor E5600 series and E5-2600 v2 series.

Device monitoring and statistics functions,such as Linux Ethtool MTU support, full RX/TX queue statistics and CRC error statistics, which enable improved system-level profiling, analysis and debug.

Support for additional Network Interface Cards(NICs), such as the Intel 82571EB Gibabit Ethernet controller, beyond those supported in the baseline Intel DPDK library.

6WIND also provides a range of optional add-on extensions to the Intel DPDK designed to improve the cost/performance of both physical and virtual networking appliances while enabling the use of the intel DPDK in software-defined networks. These optional add-ons include:

IPsec acceleration, achieved through integration of the Intel Multi-buffer Crypto for IPSec library;

Crypto acceleration via support of an external accelerator, the Intel Communications Chipset 89xx series, which is part of Intel's next-generation communications platform,codenamed "Crystal Forest"

Virtualization-related enhancements that maximize system performance by removing key I/O and communication bottlenecks include:

1. I/O Virtualization(IOV), an industry-standard approach for increasing the performance of virtual network appliances by bypassing the virtual switch within the hypervisor, thus removing the I/O performance constraints imposed by the virtual switch.

2. A virtual NIC(vNIC) driver that leverages communication between virtual machines via the virtual switch, enabling the efficient development and provisioning of systems with multiple VMs and significant East-West network traffic.

3. For system that require the ultimate level of performance for East- West traffic between VMs, a VM-to-VM driver enables direct VM-to-VM communication, bypassing the virtual switch while remaining fully compatible with industry-standard hypervisors.

These Intel DPDK enhancements and optional add-ons are maintained by 6WIND as private branch, regularly synchronized with Intel's on-going releases of the baseline library. They are delivered to customers either as a stand alone library or, for applications that also require high- performance packet processing software, and integrated within the 6WINDGate software solution.

The 6WINDGate packet processing software is designed to solve the problem of exploiting the potential packet processing performance of multicore processor through a fast pth-based architecture, while incorporating a comprehensive set of high performance networking protocols fully optimized for intel Xeon processor-based platforms.

5. Open Questions to IETF

IETF has been design Layer 2&3 protocols, and most of them are dedicated to data plane processing. The efficient implementation of protocol and tailoring them for specific hardware/chipsets have not been considered as main-stream IETF work (there are indeed some thread anyway, e.g. tailor for M2M). But to make IETF protocols as efficient as possible is definitely within the scope of IETF. Below are some discussion of open questions to IETF w.r.t. the data plane process acceleration topic.

1. Importance. The game changing initiatives already started. NFV and further virtualization and decoupling practices are

happening. Before the questions have been ported to specialized hardware, but now the industry is changing the game. Do it need the standardization collaboration?

2. Relevance. As we authors believe it, to make IETF protocols as efficient as possible is definitely within the scope of IETF. Although implementation techniques are mostly software engineering practice and have no business with any SDOs, the abstract API design and exposure of lower layer capability will definitely benefit the data plane processing efficiency.

3. Necessity. Now that DPDK is already open source. But the experience in DPDK can feedback to IETF on how to improve the protocol design in promoting data plane acceleration effectiveness.

6. Acknowledgement

This work was inspired by the DPDK open source project.

Thank you for the discussion with Hui Deng, Dapeng Liu, and Lingli Deng on how to improve and promote this document.

7. IANA Considerations

To be specified.

8. Security Considerations

TBD.

9. Informative References

[DPDK] "Packet Processing - Intel DPDK, https://01.org/packet-processing/overview/dpdk-detail", .

[NFVE2E] "Network Functions Virtualisation: End to End Architecture, http://docbox.etsi.org/ISG/NFV/70-DRAFT/0010/NFV-0010v016.zip", .

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

Authors' Addresses

Zhen Cao
China Mobile
Xuanwumenxi Ave. No. 32
Beijing 100053
China

Email: zehn.cao@gmail.com, caozhen@chinamobile.com

Qiao Fu
China Mobile
Xuanwumenxi Ave. No. 32
Beijing 100053
China

Email: fuqiao@chinamobile.com

Lingli Deng
China Mobile
Xuanwumenxi Ave. No. 32
Beijing 100053
China

Email: denglingli@chinamobile.com

DNS Extensions
INTERNET-DRAFT
Updates RFC 2845 (if approved)
Intended Status: Standards Track

Expires: August 15, 2014

H. Rafiee
Ciber AG
M. v. Loewis
C. Meinel
Hasso Plattner Institute

February 15, 2014

Secure DNS Authentication using CGA/SSAS Algorithm in IPv6
<draft-rafiee-intarea-cga-tsig-07.txt>

Abstract

This document describes a new mechanism that can be used to reduce the need for human intervention during DNS authentication and secure DNS authentication in various scenarios such as the DNS authentication of resolvers to stub resolvers, authentication during zone transfers, authentication of root DNS servers to recursive DNS servers, and authentication during the FQDN (RFC 4703) update.

Especially in the last scenario, i.e., FQDN, if the node uses the Neighbor Discovery Protocol (NDP) (RFC 4861, RFC 4862), unlike the Dynamic Host Configuration Protocol (DHCP) (RFC 3315), the node has no way of updating his FQDN records on the DNS and has no means for a secure authentication with the DNS server. While this is a major problem in NDP-enabled networks, this is a minor problem in DHCPv6. This is because the DHCP server updates the FQDN records on behalf of the nodes on the network. This document also introduces a possible algorithm for DNS data confidentiality.

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 August 15, 2014.

Table of Contents

1. Introduction

Transaction SIGnature (TSIG) [RFC2845] is a protocol that provides endpoint authentication and data integrity through the use of one-way hashing and shared secret keys in order to establish a trust relationship between two/group of hosts, which can be either a client and a server, or two servers. The TSIG keys, which are manually exchanged between a group of hosts, need to be maintained in a secure manner. This protocol is today mostly used to secure a Dynamic Update, or to give assurance to the slave name server that the zone transfer is from the original master name server and that it has not been spoofed by hackers. It does this by verifying the signature using a cryptographic key that is shared with the receiver.

But, handling this shared secret in a secure manner and exchanging it, does not seem to be easy. This is especially true if the IP addresses are dynamic due to privacy reasons or the shared secret is exposed to attacker. To address the existing problems with TSIG, this document proposes the use of Cryptographically Generated Addresses (CGA) [RFC3972] or Secure Simple Addressing Scheme for IPv6 Authoconfiguration (SSAS) as a new algorithm in the TSIG Resource Record (RR). CGA is an important option available in Secure Neighbor Discovery (SeND) [RFC3971], which provides nodes with the necessary proof of IP address ownership by providing a cryptographic binding between a host?s public key and its IP address without the need for the introduction of a new infrastructure.

This document also addresses the DNS data confidentiality by using both asymmetric and symmetric cryptography as well as data integrity. This document updates the following sections in TSIG document

- section 4.2: The server MUST not generate a signed response to an unsigned request => The server MUST not generate a signed response to an unsigned request, unless the Algorithm Name filed contains CGA-TSIG.

- Section 4.5.2: It MUST include the client’s current time in the time signed field, the server’s current time (a u_int48_t) in the other data field, and 6 in the other data length field => It MUST include the client’s current time in the time signed field, the server’s current time (a u_int48_t) in the other data field, and if the Algorithm Name is CGA-TSIG, then add the length of this client?s current time to the total length of Other DATA field. The client?s current time in this case will be placed after the CGA-TSIG Data.

1.1. Problem Statement

The authentication during any DNS query process is solely based on the source IP address when no secure mechanism is in use either

during the DNS update (zone transfer, FQDN update) or during the DNS query resolving process. This makes the DNS query process vulnerable to several types of spoofing attacks -- man in the middle, source IP spoofing, etc. One example is the problem that exists between a client and a DNS resolver. When a client sends a DNS query to a resolver, an attacker can send a response to this client containing the spoofed source IP address for this resolver. The client checks the resolver's source IP address for authentication. If the attacker spoofed the resolver's IP address, and if the attacker responds faster than the legitimate resolver, then the client's cache will be updated with the attacker's response. The client does not have any way to authenticate the resolver.

If DNSSEC (RFC 6840) or TSIG, as a security mechanism is in use, then the problem would be the manual step required for the configuration. For instance, when a DNSSEC needs to sign the zone offline. The public key verification in DNSSEC creates chicken and eggs situation. In other words, the key for verifying messages should be obtained from DNSSEC server itself. This is why the query requestor needed to ask other DNS servers up to top level in root to be able to verify the key. If this does not happen, DNSSEC is vulnerable to IP spoofing attack. This problem could easily be handled by the use of CGA-TSIG as a means of providing the proof of IP address ownership.

If TSIG is in use, the shared secret exchange is done offline. Currently there is little deployment of TSIG for resolver authentication with clients. One reason is that resolvers respond to anonymous queries and can be located in any part of the network. A second reason is that the manual TSIG process makes it difficult to configure each new client with the shared secret of the resolver. Another catastrophic problem with TSIG would be when this shared secret, that is shared between a group of hosts, leaks and makes it necessary to repeat this manual step. The reason is, that for each group of hosts there needs to be one shared secret and the administrator will need to manually add it to the DNS configuration file for each of these hosts. This manual process will need to be invoked in the case where one of these hosts is compromised and the shared secret is well known to the attacker. It will also have to be invoked in the case where any of these hosts needs to change their IP addresses, because of different reasons such as privacy issues, as explained in RFC 4941 [RFC4941], or when moving to another subnet within the same network, etc. Therefore, the problem that exists today with the authentication processes used in different scenarios is what this document addresses. The various scenarios include authentication during zone transfer, authentication of the nodes during DNS query resolving and authentication during updating PTR and FQDN (RFC 4703).

2. 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.

=> This sign in the document should be interpreted as "change to".

3. Terminology

The terms used in this document have the following standard meaning:

- Name server: A server that supports DNS service.

- Resolver/recursive DNS server: A resolver/recursive name server responds to queries where the query does not contain an entry for the node in its database. It first checks its own records and cache for the answer to the query and then, if it cannot find an answer there, it may recursively query name servers higher up in the hierarchy and then pass the response back to the originator of the query. This is known as a recursive query or recursive lookup.

- Stub resolver: A specific kind of DNS resolver that is unable to resolve the queries recursively. So, it relies on a recursive DNS resolver to resolve the queries.

- Authoritative: An authoritative name server provides the answers to DNS queries. For example, it would respond to a query about a mail server IP address or website IP address. It provides original, first-hand, definitive answers (authoritative answers) to DNS queries. It does not provide ’just cached’ answers that were obtained from another name server. Therefore it only returns answers to queries about domain names that are installed in its system configuration.

There are two types of Authoritative Name Servers:

1. Master server (primary name server): A master server stores the original master copies of all zone records. A host master is only allowed to change the master server?s zone records. Each slave server gets updated via a special automatic updating mechanism within the DNS protocol. All slave servers maintain identical copies of the master records.

2. Slave server (secondary name server): A slave server is an exact replica of the master server. It is used to share the DNS server’s load and to improve DNS zone availability in cases where the master server fails. It is recommended that there be at least 2 slave servers and one master server for each domain name.

- Root DNS server: An authoritative DNS server for a specific root domain. For example, .com

- Client: a client can be any computer (server, laptop, etc) that only supports stub DNS servers and not other DNS services. It can be a mail server, web server or a laptop computer.

- Node: a node can be anything such as a client, a DNS server (resolver, authoritative) or a router.

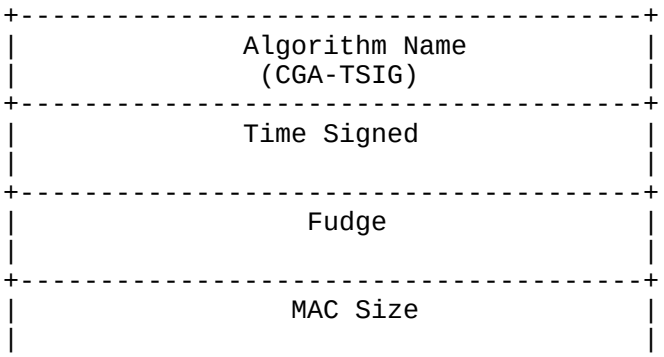
- Host: all nodes except routers

4. Algorithm overview

The following sections explain the use of CGA or any other future algorithm in place of CGA for securing the DNS process by adding a CGA-TSIG data structure as an option to the TSIG Resource Record (RR).

4.1. The CGA-TSIG DATA structure

The CGA-TSIG data structure SHOULD be added to the Other DATA section of the RDATA field in the TSIG Resource Record (RR) (see figures 1 and 2). The DNS RRTYPE MUST be set to TSIG [RFC2845]. The RDATA Algorithm Name MUST be set to CGA-TSIG. The Name MUST be set to root (.).This is the smallest possible value that can be used. The MAC Size MUST be set to 0. A detailed explanation of the standard RDATA fields can be found in section 2.3 RFC 2845. This document focuses only on the new structure added to the Other DATA section. These new fields are CGA-TSIG Len and CGA-TSIG DATA. The TSIG RR is added to an additional section of the DNS message. If another algorithm is used in place of CGA for SeND, such as SSAS [4 , 5], then the CGA-TSIG Len will be the length for the parameters of this algorithm and CGA-TSIG DATA will consist of the parameters required for verification of that algorithm, like signature, public key, etc.



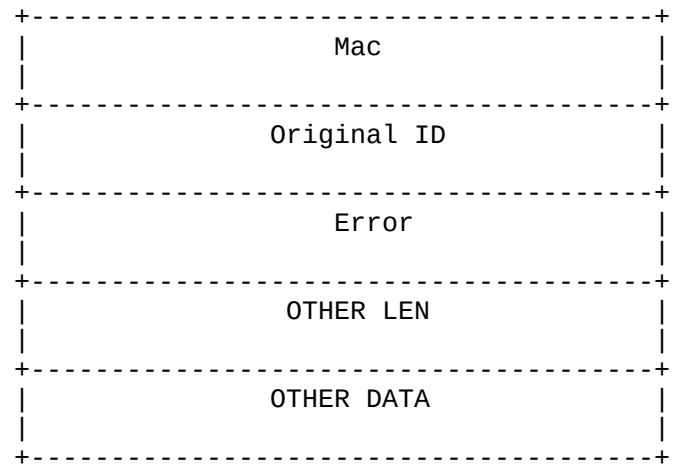


Figure 1 Modified TSIG RDATA

The CGA-TSIG DATA Field and the CGA-TSIG Len will occupy the first two slots of Other DATA. Figure 2 shows the layout. Any extra options/data should be placed after CGA-TSIG field. CGA-TSIG Len is the length of CGA-TSIG DATA in byte. This value is multiple of 8.

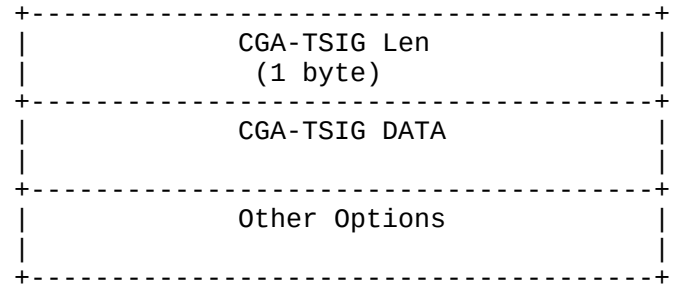


Figure 2 Other DATA section of RDATA field

CGA-TSIG DATA Field Name	Data Type	Notes
Algorithm type	u_int16_t	IANA numeric value of the algorithm for RSA 1.2.840.113549.1.1.1
type	u_int16_t	Name of the algorithm used in SEND
IP tag	16 octet	the tag used to identify the IP address
Parameters Len	Octet	the length of CGA parameters
Parameters	variable	CGA parameters Section 3 RFC 3972
Signature Len	Octet	the length of CGA signature
Signature	variable	Section 3.2.1 This document
old pubkey Len	variable	the length of old public key

		field
old pubkey	variable	Old public key in ASN.1 DER format (the same format as public key)
old Signature Len	variable	the length of old signature field
old Signature	variable	Old signature generated by old public key.

Type indicates the Interface ID generation algorithm that was used in SeND (An Interface ID is the 64 leftmost bits of an IPv6 address.). This field allows for the use of future, optional algorithms in SeND. The default value for CGA is 1. The IP tag is a node's old IP address. A client's public key can be associated with several IP addresses on a server. The DNS server, or the DNS message verifier node, SHOULD store the IP addresses and the public keys so as to indicate their association to each other. If a client wants to add RRs to the server by using a new IP address, then the IP tag field will be set to binary zeroes. The server will then store the new IP address that was passed to it in storage. If the client wants to replace an existing IP address in a DNS server with a new one, then the IP tag field will be populated with the IP address which is to be replaced. The DNS server will then look for the IP address referenced by the IP tag stored in its storage and replace that IP address with the new one. This enables the client to update his own RRs using multiple IP addresses while, at the same time, giving him the ability to change IP addresses. If a node changes its public key in order to maintain privacy, then it MUST add the old public key to the old pubkey field. It MUST also retrieve the current time from Time Signed field, sign it using the old private key, and then add the digest (signature) to the old signature field. This enables the verifier node to authenticate a host with a new public key. The detailed verification steps are explained in sections 5.1, 6.1 and 7.1.

4.2. Generation of CGA-TSIG DATA

In order to use CGA-TSIG as an authentication approach, some of the parameters need to be cached during IP address generation. If no parameters are available in cache, please see section 8. If the Type (section 4.1) is CGA, then the parameters that SHOULD be cached are the modifier, algorithm type, location of the public/private keys and the IP addresses of this host generated by the use of CGA.

1. Obtain required parameters from cache.

The CGA-TSIG algorithm obtains the old IP address, modifier, subnet prefix, collision count and public key from cache. It concatenates the old IP address with the CGA parameters, i.e., modifier, subnet prefix, collision count, public key (the order of CGA parameters are shown in section 3 RFC 3972). If the old IP address is not available, then CGA-TSIG must set the old IP address (IP tag) to zero.

Note: If the node is a DNS server (resolver or authoritative DNS server) which does not support SeND, but wants to use CGA-TSIG algorithm, then it is possible to use a script to generate the CGA parameters, which are needed to manually configure this server's IP address. Then this server can make use these parameters for authentication purposes.

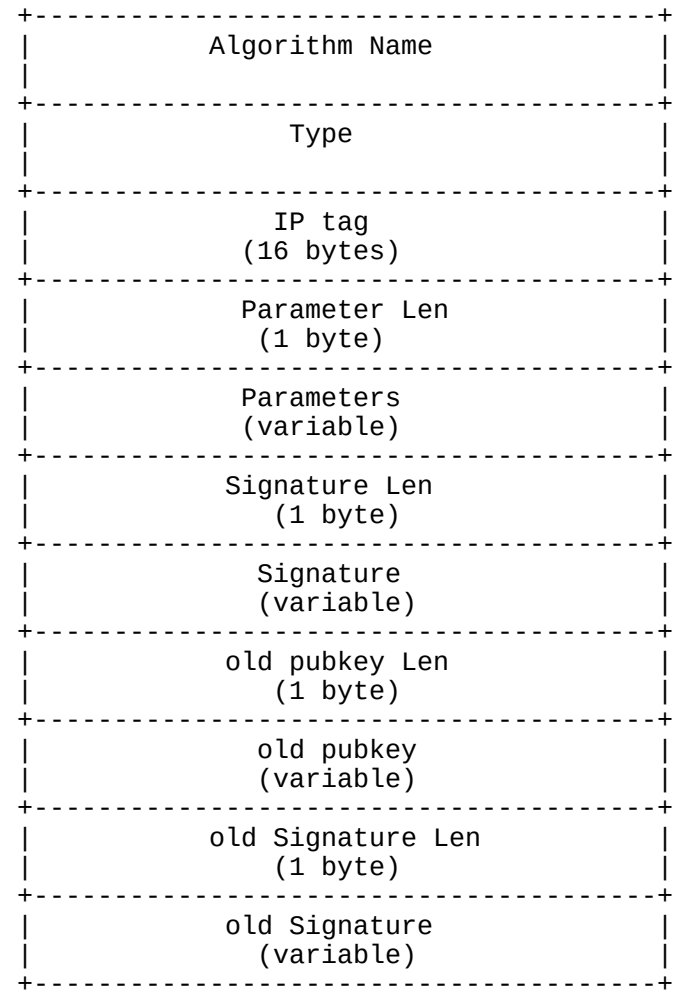


Figure 3 CGA-TSIG DATA Field

2. Generate signature

For signature generation, The 128-bit CGA Message Type tag value for SeND that is 0x086F CA5E 10B2 00C9 9C8C E001 6427 7C08, is concatenated with the whole DNS message from Type to additional data sections (Please refer to figure 4 and figure 5) excluding the signature fields itself in the CGA-TSIG DATA is signed by using a RSA algorithm, by default, or any future algorithm used in place of RSA,

and the private key which was obtained from cache in the first step. This signature must be added to the signature field of the CGA-TSIG DATA. Time Signed is the same timestamp as is used in RDATA. This value is the number of seconds since 1 January 1970 in UTC 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 message. The format of DNS messages is explained in section 4.1.3 RFC 1035 [RFC1035]. Figure 6 shows this signature.

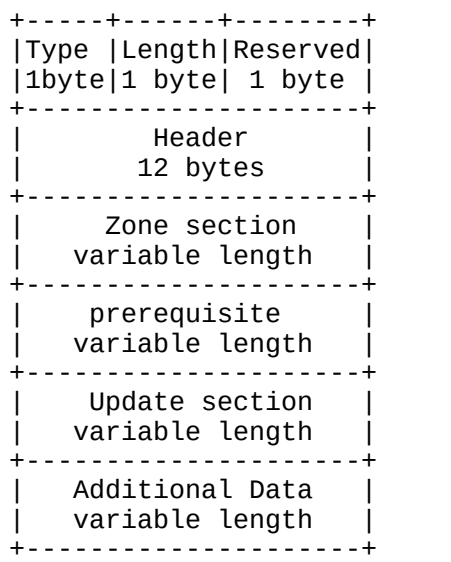


Figure 4 DNS update message

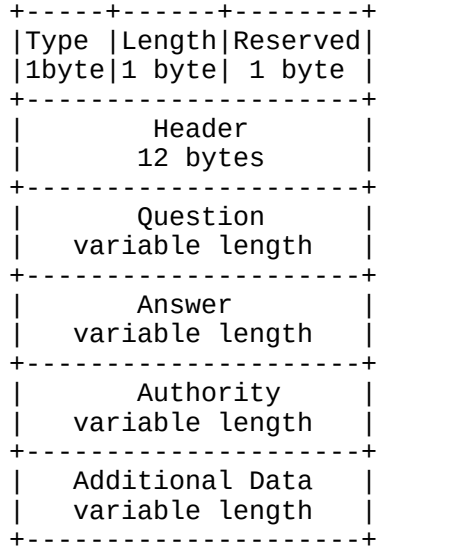


Figure 5 DNS Query message (section 4.1 RFC 1035)

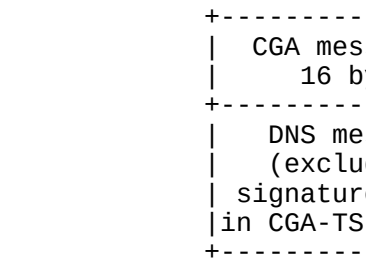


Figure 6 CGA-TSIG Signature content

3. Generate old signature

If the nodes generated new key pairs, then they need to add the old public key and message, signed by the old private key, to CGA-TSIG DATA. A node will retrieve the timestamp from Time Signed, will use the old private key to sign it, and then will add the content of this signature to the old signature field of CGA-TSIG DATA. This step MUST be skipped when the node did not generate new key pairs.

5. Authentication during Zone Transfer

This section discusses the use of CGA-TSIG for the authentication of two DNS servers (a master and a slave). In the case of processing a DNS update for multiple DNS servers (authentication of two DNS servers), there are two possible scenarios with regard to the authentication process, which differs from that of the authentication of a node (client) with one DNS server. This is because of the need for human intervention.

a. Add the 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 use of the CGA-TSIG option from the TSIG RR. This scenario will be valid until the IP address in any of these DNS servers, changes.

To automate this process, the sender’s public key of the DNS Update message must 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. 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, the certificate should be saved on the DNS server for later use for the generation of its address or for the DNS update process. In this case, whenever any of these servers want to generate a new IP address, then the DNS update process can be accomplished automatically without the need for human intervention.

5.1. Verification process

Sender authentication is necessary in order to prevent attackers from making unauthorized modifications to DNS servers through the use of spoofed DNS messages. The verification process executes the following steps:

1. Verify the signature

The signature contained in CGA-TSIG DATA should be verified. This can be done by retrieving the public key and signature from CGA-TSIG DATA and using this public key to verify the signature. If the verification process is successful, then step 2 will be executed. If the verification fails, then the message should be discarded without further action.

2. Check the Time Signed

The Time Signed value is obtained from TSIG RDATA and is called t1. The current system time is then obtained and converted to UTC time and is called t2. Fudge time is obtained from TSIG RDATA. If t1 is in the range of t2 and t2 minus/plus fudge (see formula 1) then step 3 will be executed. Otherwise, the message will be considered a spoofed message and the message should be discarded without further action. The range is used in consideration of the delays that can occur during its transmission over TCP or UDP. Both times must use UTC time in order to avoid differences in time based on different geographical locations.

(t1 - fudge) <= t2 <=(t1 + fudge) (1)

3. Execute the CGA verification

These steps are found in section 5 RFC 3972. If the sender of the DNS message uses another algorithm, instead of CGA, then this step becomes the verification step for that algorithm. If the verification process is successful, then step 4 will be executed. Otherwise the message will be discarded without further action.

4. Verify the source IP address

The source IP address of the Update requester MUST be checked against the one contained in the DNS configuration file. If it is the same, then the Update Message should be processed, otherwise, step 5 will be executed.

5. Verify the public key

The DNS server checks whether or not the public key retrieved from CGA-TSIG DATA is the same as what was available in the storage where the public keys and IP addresses were saved. If no entry is found in storage for this public key, then the update will be rejected without further action. Otherwise, when the old public key length is not zero go to step 6.

6. Verify the old public key

If the old public key length is zero, then skip this step and discard the DNS update message without further action. If the old public key length is not zero, then the DNS server will retrieve the old public key from CGA-TSIG DATA and will check to see whether or not it is the same as what was saved in the DNS server’s storage where the public keys and IP addresses are stored. If it is the same, then step 6 will be executed, otherwise the message should be discarded without further action.

7. Verify the old signature

The old signature contained in CGA-TSIG DATA should be verified. This can be done by retrieving the old public key and the old signature from CGA-TSIG DATA and then using this old public key to verify the old signature. If the verification is successful, then the Update Message should be processed and the new public key should be replaced with the old public key in the DNS server. If the verification process fails, then the message should be discarded without further action.

6. Authentication during the FQDN or PTR Update

Normally the DHCPv6 server will update the client’s RRs on their behalf in the scenario where SeND is used as a secure NDP, the nodes will need to do this process themselves unless there is stateless DHCPv6 server available. CGA-TSIG can be used to give nodes the ability of doing this process themselves. In this case the clients need to include the CGA-TSIG option in order to allow the DNS server to verify them. The verification process is the same as that explained in section except for step 4.

6.1. Verification Process

The verification steps are the same as those is explained in section 5.1, but removing step 4 and modifying step 5.

1- Verify the signature

2- Check the Time Signed

3- Execute the CGA verification

4. Verify the public key

The DNS server checks whether or not the public key retrieved from CGA-TSIG DATA is the same as what was available in the storage where the public keys and IP addresses were saved. If no entry is found in storage for this public key, and the FQDN or PTR is also not available in the DNS server, then the DNS server will store the public key of this node in his database and add this node's PTR and FQDN. Otherwise if any PTR is available, and the node IP tag is empty, or there is currently another public key associated with the node's FQDN, then the update will be rejected without further action. Otherwise go to step 5 when the old public key length is not zero.

5- Verify the public key

6- Verify the old public key

7- Verify the old signature

7. Authentication during Query Resolving (stub to recursive)

A DNS query request sent by a host, such as a client or a mail server, does not need to generate CGA-TSIG DATA because the resolver responds to anonymous queries. But the resolver's response SHOULD contain the CGA-TSIG DATA field in order to enable this client to verify him. However, the client needs to include the TSIG RDATA and set the Algorithm type to CGA-TSIG. It MUST set the CGA-TSIG Len to zero. This allows the resolver to know when to include CGA-TSIG for verification process in client.

In generation of the CGA-TSIG for a resolver, there is no need to include the IP tag. This is because resolvers do not usually have several IP addresses so the client does not need to keep several IP addresses for the same resolver.

7.1. Verification process

When a resolver responds to the host's query request for the first time, the client saves its public key in a file. This allows the client to verify this resolver when it changes its IP address due to privacy or security concerns. The steps 2 and 3 of the verification process are the same as those steps explained in section 5.1. These steps are as follows:

1. Verify the signature

The signature contained in CGA-TSIG DATA should be verified. This can be done by retrieving the public key and signature from CGA-TSIG DATA and using this public key to verify the signature. If the verification process is successful, then step 2 will be executed. If the verification fails, then the message should be discarded without further action.

2. Check the Time Signed

3. Execute the CGA verification

4. Verify the Source IP address

If the resolver's source IP address is the same as that which is known for the host, then step 5 will be executed. Otherwise the message SHOULD be discarded without further action.

5. Verify the public key

The host checks whether or not the public key retrieved from CGA-TSIG DATA matches any public key that was previously saved in the storage where the public keys and IP addresses of resolvers are saved. If there is a match, then the message is processed. If not, then step 5 will be executed.

5. Verify the old public key

If the old public key length is zero, then skip this step and discard the DNS query response without further action. If the old public key length is not zero, then the host will retrieve the old public key from CGA-TSIG DATA and will check whether or not it is the same as what was saved in the host's storage where the public keys and IP addresses are stored. If it is the same, then step 6 will be executed, otherwise the message should be discarded without further action.

6. Verify the old signature

The old signature contained in CGA-TSIG DATA should be verified. This can be done by retrieving the old public key and old signature from CGA-TSIG DATA and then using this old public key to verify the old signature. If the verification is successful, then the DNS Message should be processed and the new public key should be replaced with the old public key of the resolver in the host. If the verification

process fails, then the message should be discarded without further action.

8. Authentication during Query Resolving (Auth. to recursive)

This verification step in the authentication of authoritative to recursive DNS server is the same as that explained in section 7.1. In this case the recursive DNS server does not need to generate CGA-TSIG DATA, but the root DNS server does need to include it in order to enable the recursive DNS server to verify it. The recursive DNS server needs to include the TSIG RDATA and set the Algorithm type to CGA-TSIG. It MUST set the CGA-TSIG Len to zero. This allows the root DNS server to know when to include CGA-TSIG for verification process in client.

9. No cache parameters available or SeND is not supported

In the case where there are no cache parameters available during the IP address generation, there are then two scenarios that come into play here. In the first scenario there is the case where the sender of a DNS message needs to generate a key pair and generate the CGA-TSIG data structure as explained in section 4. The node SHOULD skip the first section of the verification processes explained in section 5.1 , section 6.1 and section 7.1.

In the second scenario, as explained in section 4.2 (step 1), it is not necessary for the server to support the SeND or CGA algorithm. The DNS administrator can make a one-time use of a CGA script to generate the CGA parameters and then manually configure the IP address of this DNS server. Then later, this DNS server can use those values as a means for authenticating other nodes. The verifier nodes also do not necessarily need to support SeND. They only need to support CGA-TSIG.

10. How to obtain the IP address of resolvers

Nodes can obtain the IP address of resolvers from the DHCPv6 server (that will not be secure) or from a DNS option of Router Advertisement message [RFC6106] after authenticating the router via a trusted authority. The IP addresses can be generated using CGA, SSAS or other mechanisms.

11. CGA-TSIG Data confidentiality

One possible solution to provide the DNS server with data

confidentiality during DNS update or other DNS query processes is the use of symmetric encryption with CGA-TSIG that is called CGA-TSIGe. In this case, the node MUST set the Algorithm type in TSIG RDATA to CGA-TSIGe.

11.1. Generation of secret key

To encrypt the DNS message using a symmetric algorithm for performance purposes, first, a node needs to retrieve the public key of the DNS server. It is possible to use the current DNSKEY RR (RFC 3757) to send the public key of the DNS server. When the client wants to update any records on the DNS server, it first sends a DNS message and asks for the public key of the DNS server. DNS server then answers to this query and includes the public key contained in the DNSKEY RR with the SEP flag set to zero. This is done to indicate that it is not the zone key. The DNS server SHOULD include CGA-TSIG DATA so that the client can verify its IP address. In this case, there will be a binding between DNS server?s public key and its IP address. After a successful verification, the node then generates a 16 byte random number and calls it a secret key. It encrypts this secret key using the DNS server public key. This allows only the DNS server to decrypt this secret key. In this case, the node sets the MAC in TSIG RDATA to the digest of secret key and set the MAC Size to the length of this digest. The DNS server knows what to do with MAC field from the Algorithm type in TSIG. If it is CGA-TSIGe, then it looks for an encrypted secret key.

11.2. DNS message generation

The node MUST encrypt all DNS message sections that required protections using the secret key generated in last section and AES symmetric algorithm. It excludes TSIG RDATA (That usually added in the additional section of the DNS messages) from the encryption text. They are explained in figure 4 and figure 5 of section 4.2 of this document.

11.3. CGA-TSIGe DATA generation

The CGA-TSIGe generation is the same as that explained in section 4.2 of this document. But only the Algorithm type MUST be set to CGA-TSIGe.

11.4. Process of encrypted DNS message

When the DNS server receives the message from any node with TSIG

RDATA Algorithm type set to CGA-TSIGe, it execute the following steps:

1- Retrieve the secret key

The DNS server retrieves the secret key from MAC field. It then decrypts this secret key using its own private key.

2- Decrypt the DNS message

The DNS server decrypts the DNS server message using this secret key and the symmetric algorithm, which by default is AES.

Then the DNS server can starts the verification process as explained in section 5.1, 6.1, 7.1 of this document.

12. CGA-TSIG/CGA-TSIGe Applications

The purpose of CGA-TSIG [7] is to minimize the amount of human intervention required to accomplish shared secret or key exchange and, as a byproduct, to reduce the process's vulnerability to attacks introduced by human errors (during changing the DNS configuration) when Secure Neighbor Discovery (SeND) is used for addressing purposes or when SeND is not available for use.

As explained in a prior section, CGA-TSIG can be used in different scenarios. For the FQDN update scenario CGA-TSIG is useful in dynamic networks where the nodes want to change their IP addresses frequently in order to maintain privacy. If the Dynamic Host Configuration Protocol (DHCP) is in use, then the DHCP server can do this update on behalf of the nodes in this network on a DNS server but in Neighbor Discovery Protocol (NDP), there is no feature available that allows the host security update process for its own FQDN. CGA-TSIG can be a solution.

For the resolver scenario, usually the resolver can add the TSIG Resource Record (RR) to the DNS query response and use the CGA-TSIG algorithm in order to permit a useful authentication of the result. CGA-TSIG assures the client that the query response comes from the true originator and not from an attacker. It also ensures the integrity of the data by signing the data.

There are several types of attack that CGA-TSIG can prevent. Here we will evaluate some of them. The use of CGA-TSIG will also reduce the number of messages needed in exchange between a client and a server in order to establish a secure channel. To exchange the shared secret between a DNS resolver and a client, when TSIG is used, a minimum of four messages are required for the establishment of a secure channel. Modifying RFC 2845 to use CGA-TSIG will decrease the number of messages needed in this exchange. The messages used in RFC 2930 (TKEY RR) are not needed when CGA-TSIG is used.

12.1. IP Spoofing

During the DNS Update process or the query resolving process it is important that both communicating parties know that the one that they are communicating with is the actual owner of that IP address and that the messages are not being sent from a spoofed IP address. This can be accomplished by the use of the CGA algorithm which utilizes the node for IP address verification of other nodes.

12.2. DNS Dynamic Update Spoofing

Dynamic Update Spoofing is eliminated because the signature contains both the CGA parameters and the DNS update message. This will offer proof of the sender’s IP address ownership (CGA parameters) and the validity of the update message.

12.3. Resolver Configuration Attack

When using CGA-TSIG, the DNS server, or the client, would not need further configuration. This would reduce the possibility of human errors being introduced into the DNS configuration file. Since this type of attack is predicated on human error, the chances of it occurring, when this extension is used, are minimized.

12.4. Exposing Shared Secret

Using CGA-TSIG will decrease the number of manual steps required in generating the new shared secret and in exchanging it among the hosts where the old shared secret was shared between them for updating purposes. This manual step is required after a leakage has occurred of the shared secret to an attacker via any of these hosts.

12.5. Replay attack

Using the Time Signed value in the signature modifies the content of the signature each time the node generates and sends it to the DNS server. 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 the signature. In this case, the verification process will fail thus also preventing the replay attack.

12.6. Data confidentiality

Encrypting the whole DNS message will avoid the attacker to know the content of DNS messages. This will avoid zone walking and many other attacks on DNS RRs. This also provides the higher privacy for hosts that has DNS records.

13. Security Considerations

The approach explained in this draft, CGA-TSIG, is a solution for securing DNS messages from spoofing type attacks like those explained in section 3.

A problem that may arise here concerns attacks against the CGA algorithm. In this section we will explain the possibility of such attacks against CGA [5] and explain the available solutions that we considered in this draft.

a) Discover an Alternative Key Pair Hashing of the Victim’s Node Address

In this case an attacker would have to find an alternate key pair hashing of the victim?s address. The probability for success of this type of attack will rely on the security properties of the underlying hash function, i.e., an attacker will need to break the second pre-image resistance of that hash function. The attacker will perform a second pre-image attack on a specific address in order to match other CGA parameters using Hash1 and Hash2. The cost of doing this is (2^59+1) * 2^(16*1). If the user uses a sufficient security level, it will be not feasible for an attacker to carry out this type of attack due to the cost involved. Changing the IP address frequently will also decrease the chance for this type of attack succeeding.

b) DoS to Kill a CGA Node

Sending a valid or invalid CGA signed message with high frequency across the network can keep the destination node(s) busy with the

verification process. This type of DoS attack is not specific to CGA, but it can be applied to any request-response protocol. One possible solution ,to mitigate this attack, is to add a controller to the verifier side of the process to determine how many messages a node has received over a certain period of time from a specific node. If a determined threshold rate is exceeded, then the node will stop further receipt of incoming messages from that node.

c) CGA Privacy Implication

Due to the high computational complexity necessary for the creation of a CGA, it is likely that once a node generates an acceptable CGA it will continue its use at that subnet. The result is that nodes using CGAs are still susceptible to privacy related attacks. One solution to these types of attacks is setting a lifetime for the address as explained in RFC 4941.

14. 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.

In section 4.1, Type should allow for the use of future optional algorithms with regard to SeND. The default value for CGA might be 1. Other algorithms would be assigned a new number sequentially. For example, a new algorithm called SSAS [4,5] could be assigned a value of 2.

IANA also needs to define a numeric algorithm number for ECC. The similar way that is defined for RSA.

15. Appendix

- A sample key storage for CGA-TSIG

```
create table cgatsigkeys (

id           INT auto_increment,

pubkey       VARCHAR(300),

primary key(id)

);
```

```
create table cgatsigips (

id           INT auto_increment,

idkey                INT,

IP           VARCHAR(20),

FOREIGN KEY (idkey) REFERENCES cgatsigkeys(id)

primary key(id)

);
```

CGA-TSIG tables on mysql backend database

- a sample format of stored parameters in the node

For example, the modifier is stored as bytes and each byte might be separated by a comma (for example : 284,25,14,...). Algorithmtype is the algorithm used in signing the message. Zero is the default algorithm for RSA. Secval is the CGA Sec value that is, by default, one. GIP is the global IP address of this node (for example: 2001:abc:def:1234:567:89a). oGIP is the old IP address of this node, before the generation of the new IP address. Keys contains the path where the CGA-TSIG algorithm can find the PEM format used for the public/private keys (for example: /home/myuser/keys.pem).

```
<?xml version="1.0" encoding="UTF-8"?>

<Details>

<CGATSIG>

<modifier value=""/>

<algorithmtype value="1.2.840.113549.1.1.1"/>

<secval value="1"/>

<GIP value=""/>

<oGIP value=""/>

<Keys value=""/>

</CGATSIG>

</Details>
```

XML file contains the cached DATA

16. Acknowledgements

The continual improvement of this document is as a result of the helps and assistance of its supporters.

The authors would like to thank all those people who directly helped in improving this draft and all supporters of this draft, especially Ralph Droms, Andrew Sullivan, Ted Lemon, Brian Haberman. The authors would like also to special acknowledge the supports of NLnet Labs director and researchers; Olaf Kolkman, Matthijs Mekking and their master student Marc Buijsman.

17. References

17.1. Normative References

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

[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.

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

[RFC2136] Vixie, P. (Editor), Thomson, S., Rekhter, Y., Bound, J., "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2845] Vixie, P., Gudmundsson, O. , Eastlake 3rd, D., Wellington, B., " Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC6106] Jeong, J., Park, S., Beloeil, L., Madanapalli, S.,"IPv6 Router Advertisement Options for DNS

Configuration",RFC 6106, November 2010.

17.2. 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.

[3] AlSa'deh, A., Rafiee, H., Meinel, C., "IPv6 Stateless Address Autoconfiguration: Balancing Between Security, Privacy and Usability". Lecture Notes in Computer Science, Springer(5th International Symposium on Foundations & Practice of Security (FPS). October 25 - 26, 2012 Montreal, QC, Canada), 2012.

[4] Rafiee, H., Meinel, C., "A Simple Secure Addressing Generation Scheme for IPv6 AutoConfiguration (SSAS)". Work in progress, http://tools.ietf.org/html/draft-rafiee-6man-ssas, 2013.

[5] Rafiee, H., Meinel, C., "A Simple Secure Addressing Scheme for IPv6 AutoConfiguration (SSAS)", 11th International conference on Privacy, Security and Trust (IEEE PST), 2013.

[6] AlSa'deh, A., Rafiee, H., Meinel, C., "Cryptographically Generated Addresses (CGAs): Possible Attacks and Proposed Mitigation Approaches," in proceedings of 12th IEEE International Conference on Computer and Information Technology (IEEE CIT'12), pp.332-339, 2012.

[7] Rafiee, H., Meinel, C., "A Secure, Flexible Framework for DNS Authentication in IPv6 Autoconfiguration" in proceedings of The 12th IEEE International Symposium on Network Computing and Applications (IEEE NCA13), 2013.

Authors' Addresses

Hosnieh Rafiee
Ciber AG
KoelnTurm
Im Mediapark 8
50670, Cologne
http://www.ciber.com
Phone: +49 (0221) 272 67- 122
Email: ietf@rozanak.com

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

Martin von Loewis
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany