

INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: August 2008

Donald E. Eastlake 3rd
Motorola Laboratories
February 25, 2008

Domain Name System (DNS) Cookies

[<draft-eastlake-dnsext-cookies-03.txt>](#)

Status of This Document

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

This draft is intended to become a Proposed Standard RFC. Distribution of this document is unlimited. Comments should be sent to the author or the DNSEXT working group mailing list [<namedroppers@ops.ietf.org>](mailto:namedroppers@ops.ietf.org).

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/lid-abstracts.html>

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

Abstract

DNS cookies are a light-weight DNS transaction security mechanism designed for incremental deployment. They provide limited protection to DNS servers and resolvers against a variety of increasingly common denial-of-service and cache poisoning or forgery attacks by off-path attackers. DNS Cookies are tolerant of NAT, NAT-PT, and Anycast.

Table of Contents

Status of This Document.....[1](#)
Abstract.....[1](#)

Table of Contents.....[2](#)

[1](#). Introduction.....[3](#)
[1.1](#) Contents of This Document.....[3](#)
[1.2](#) Definitions.....[4](#)

[2](#). Threats Considered.....[5](#)
[2.1](#) Denial-of-Service Attacks.....[5](#)
[2.1.1](#) DNS Server Denial-of-Service.....[5](#)
[2.1.2](#) Selected Host Denial-of-Service.....[5](#)
[2.2](#) Cache Poisoning and Answer Forgery Attacks.....[6](#)
[3](#). Comments on Existing DNS Security.....[6](#)
[3.1](#) Existing DNS Data Security.....[6](#)
[3.2](#) DNS Message or Transaction Security.....[7](#)
[3.3](#) Conclusions on Existing DNS Security.....[7](#)

[4](#). The COOKIE OPT option.....[8](#)
[4.1](#) Resolver Cookies.....[8](#)
[4.2](#) Server Cookies.....[9](#)
[5](#). DNS Cookie Policies and Implementation Requirements.....[9](#)
[5.1](#) Resolver Policies and Implementation.....[10](#)
[5.2](#) Server Policies and Implementation.....[11](#)
[5.3](#) Implementation Requirements.....[11](#)
[6](#). NAT Considerations and AnyCast Server Considerations...[12](#)

[7](#). Incremental Deployment.....[14](#)

[8](#). IANA Considerations.....[15](#)
[9](#). Security Considerations.....[15](#)
[10](#). Normative References.....[16](#)
[11](#). Informative References.....[16](#)

Author's Address.....[18](#)
Copyright and Disclaimer.....[18](#)

Additional IPR Provisions.....[18](#)
Expiration and File Name.....[19](#)

[1.](#) Introduction

As with many core Internet protocols, the Domain Name System (DNS) was designed at a time when the Internet had only a small pool of trusted users. As the Internet has exploded to a global information utility, the DNS has increasingly been subject to abuse and been used as a vector for abuse.

This document describes DNS cookies, a light-weight DNS transaction security mechanism specified as an OPT [[RFC2671](#)] option. This mechanism provides limited protection to DNS servers and resolvers against a variety of increasingly common denial-of-service and cache poisoning forgery attacks by off-path attackers.

The DNS cookies mechanism is designed with a default mode which supports incremental deployment. If only one party to a DNS transaction supports the mechanism, it does not interfere or provide a benefit, but, if both support it, the additional security provided is automatically available for that and subsequent transactions.

The DNS cookies mechanism is compatible with and can be used in conjunction with other DNS transaction forgery resistance measures such as those in [[forgery](#)].

The DNS cookies mechanism is designed to work in the presence of NAT and NAT-PT boxes and guidance is provided herein on supporting the DNS cookies mechanism in anycast servers.

[1.1](#) Contents of This Document

In [Section 2](#), we discuss the threats against which the DNS cookie mechanism provides some protection.

[Section 3](#) describes existing DNS security mechanisms and why they are not adequate substitutes for DNS cookies.

[Section 4](#) describes the COOKIE OPT option including recommendations for calculating Resolver and Server Cookies.

[Section 5](#) describes the processing of COOKIE OPT options by resolvers and server and policies for such processing.

[Section 6](#) discusses some NAT and anycast related DNS Cookies design considerations.

[Section 7](#) discusses incremental deployment considerations.

Sections [8](#) and [9](#) describe IANA and Security Considerations.

[1.2](#) Definitions

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](#)].

An "off-path attacker", for a particular DNS resolver and server, is defined as an attacker which cannot observe the legitimate plain text DNS requests and responses between that resolver and server.

"Soft state" indicates information learned or derived by a host which may be discarded when indicated by the policies of that host but can be later re-instantiated if needed. For example, it could be discarded after a period of time or when storage for caching such data becomes full. If operations requiring that soft state continue after it has been discarded, it will be automatically re-generated, albeit at some cost.

"Silently discarded" indicates that there are no DNS protocol message consequences; however, it is RECOMMENDED that appropriate debugging network management facilities be included in implementations, such as a counter of the occurrences of each type of such events.

The term "IP address" is used herein in a length independent manner and refers to address formats including IPv4 and IPv6.

[2.](#) Threats Considered

DNS cookies are intended to provide significant but limited protection against certain denial-of-service and cache poisoning or answer forgery attacks by off-path attackers described below.

[2.1](#) Denial-of-Service Attacks

The canonical form of the denial-of-service attacks considered herein is to send DNS requests with forged source IP addresses to a server. The intent can be to attack that server or a selected host as described below.

2.1.1 DNS Server Denial-of-Service

DNS requests that are accepted cause work on the part of DNS servers. This is particularly true for recursive servers which may issue one or more requests and process the responses thereto in order to determine their response to the initial query. And the situation is even worse for recursive servers implementing DNSSEC [[RFC4033](#)], [[RFC4034](#)], [[RFC4035](#)] because they may be induced to perform burdensome public key cryptographic computations in attempts to verify the authenticity of data they retrieve while trying to answer the request.

The computational or communications burden caused by such requests may not depend on a forged IP source address, but the use of such addresses makes

- + the source of the requests causing the denial-of-service attack to be harder to find and
- + administrative restriction of the IP addresses from which such requests should be honored harder or impossible to specify.

Use of DNS cookies almost always enables a server to reject forged queries from an off path attacker with relative ease, certainly before any recursive queries or public key cryptographic operations are performed.

2.1.2 Selected Host Denial-of-Service

A request with a forged IP address generally causes a response to be sent to that forged IP address. Thus the forging of many such requests with a particular source IP address can result in enough traffic being sent to the forged IP address to interfere with service

to the host at the IP address. Furthermore, it is generally easy in the DNS to create short requests that produce much longer responses. Thus a DNS server can be used as not only a way to obscure the true source of an attack but as a traffic amplifier to make the attack more effective.

Use of the DNS Cookies mechanism severely limits the traffic

amplification that can be obtained by attackers off path for the server and the attacked host. Enforced DNS cookies would make it hard for an off path attacker to cause any more than a brief error response to be sent to a forged IP address. Furthermore, DNS cookies make it more effective to implement a rate limiting scheme for bad DNS cookie error responses from the server. Such a scheme would further restrict selected host denial-of-service traffic from that server.

[2.2](#) Cache Poisoning and Answer Forgery Attacks

The form of the cache poisoning attacks considered is to send forged replies to a resolver. Modern network speeds for well connected hosts are such that, by forging replies from the IP addresses of heavily used DNS servers for popular names to a heavily used resolver, there can be an unacceptably high probability of randomly coming up with a reply that will be accepted and cause false DNS information to be cached by that resolver. This can be used to facilitate phishing attacks and other diversion of legitimate traffic to a compromised or malicious host such as a web server.

[3.](#) Comments on Existing DNS Security

Two forms of security have been added to DNS, data security and message or transaction security.

[3.1](#) Existing DNS Data Security

DNS data security is one part of DNSSEC and is described in [[RFC4033](#)], [[RFC4034](#)], and [[RFC4035](#)]. It provides data origin authentication and authenticated denial of existence. It is being deployed slowly and, in any case, can make some denial-of-service attacks worse because of the high cryptographic computational load it can require and the increased size in DNS packets that it tends to produce.

[3.2](#) DNS Message or Transaction Security

The second form of security which has been added to DNS provides "transaction" security through TSIG [[RFC2845](#)] or SIG(0) [[RFC2931](#)]. TSIG could provide near perfect protection against the attacks for which the DNS Cookies mechanism provide weak and incomplete protection; however, TSIG is hard to deploy in the general Internet because of the burden it imposes of pre-agreement and key distribution between pairs of resolvers and servers and because it requires time synchronization between resolver and server.

TKEY [[RFC2930](#)] can solve the problem of key distribution for TSIG but some modes of TKEY impose a substantial cryptographic computation loads and can be dependent on the deployment of DNSSEC.

SIG(0) provides less denial of service protection than TSIG or, in one way, even DNS cookies, because it does not authenticate requests, only complete transactions. In any case, it also depends on the deployment of DNSSEC and requires computationally burdensome public key cryptographic operations.

[3.3](#) Conclusions on Existing DNS Security

Thus, none of the previous forms of DNS security are a suitable substitute for the DNS Cookies mechanism, which provide light weight message authentication of DNS requests and responses with no requirement for pre-configuration.

INTERNET-DRAFT

DNS Cookies

4. The COOKIE OPT option

COOKIE is an OPT RR [[RFC2671](#)] option that can be included no more than once in the RDATA portion of an OPT RR in DNS requests and responses.

The option is encoded into 22 bytes as shown below.

```

          1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|   OPTION-CODE = {TBD}           |   OPTION-LENGTH = 18           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                               |
+-+                               +-+                               +-+
|                                     |                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                               |
+---+                               +---+                               +---+
|                                     |                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Error Code           |
+-----+-----+-----+-----+-----+-----+-----+

```

The 64-bit Resolver and Server Cookies are stored in network byte order and are determined as described below.

The Error Code field MUST be zero in requests and in responses unless the response is communicating a DNS cookie error. Three values are specified in this document for Error Code: NOCOOKIE and BADCOOKIE which occur with a Refused RCODE in the DNS response header, and MANYCOOKIE which occurs with a FormErr RCODE in the DNS header. More information on the generation of error responses appears in [Section 5](#) below.

4.1 Resolver Cookies

The Resolver Cookie, when it occurs in an OPT in a DNS response, is intended to weakly assure the resolver that the response came from a server at the indicated source IP address.

Servers remember the Resolver Cookie that appears in a query long enough to use it in the construction of the COOKIE OPT option in the corresponding response if such a COOKIE OPT option is included in that response.

The Resolver Cookie SHOULD be a pseudo-random function of the server IP address and a secret quantity known only to the resolver. This

resolver secret SHOULD have 64 bits of entropy [[RFC4086](#)] and MAY be changed periodically. The RECOMMENDED method is the HMAC-SHA1-64 [[RFC1321](#)] [[RFC2104](#)] of the server IP address and the resolver secret. That is

$$\text{Resolver Cookie} = \text{Truncate-64} \\ \left(\text{HMAC-SHA1} \left(\text{Server IP Address, Resolver Secret} \right) \right)$$

where Truncate-64 takes the first 64 bits. A resolver MUST NOT use the same Resolver Cookie value for queries to all servers.

[4.2](#) Server Cookies

The Server Cookie, when it occurs in a COOKIE OPT option in a query, is intended to weakly assure the server that the query legitimately came from a resolver at the indicated source IP address that is using that Resolver Cookie.

Resolvers learn Server Cookies and retain them as soft state associated with the server IP address. They learn them from the Server Cookie that appears in the COOKIE OPT option of a reply if that reply has the correct Resolver Cookie, even if that reply is an error message.

The Server Cookie SHOULD be a pseudo-random function of the request source IP address, the request Resolver Cookie, and a secret quantity known only to the server. This server secret SHOULD have 64 bits of entropy [[RFC4086](#)] and SHOULD be changed periodically such as daily. The RECOMMENDED method is the HMAC-SHA1-64 [[RFC1321](#)], [[RFC2104](#)] of the request IP address, the Resolver Cookie, and the server secret. That is

Server Cookie = Truncate-64 (HMAC-SHA1 (Request IP Address | Resolver Cookie), Server Secret))

where Truncate-64 takes the first 64 bits and "|" represents concatenation.

A server MUST NOT use the same Server Cookie value for responses to all resolvers.

[5.](#) DNS Cookie Policies and Implementation Requirements

DNS resolvers and servers will adopt one of various policies regarding cookies. These policies SHOULD be logically settable on a per server IP address basis for resolvers and a per resolver (IP

D. Eastlake 3rd

[Page 9]

INTERNET-DRAFT

DNS Cookies

address, Resolver Cookie) pair for servers. Thus a resolver can have different policies for different servers, based on the server IP address. And a server can have different policies for different resolvers, based on the resolver IP address and Resolver Cookie. Of course, the actual implementation of setting these policies may be for blocks or classes of values or use sparse array techniques or the like.

The policy for each value is either "Disabled", "Enabled", or "Enforced" as described below.

[5.1](#) Resolver Policies and Implementation

A resolver will logically have one of the following three modes of operation or "policies" for each DNS server as distinguished by server IP Address.

Disabled:

- Never include a COOKIE OPT option in requests.
- Ignore COOKIE OPT options in responses.

Enabled:

- Always include a COOKIE OPT option in requests. If a cached Server Cookie for the server is not available, the Server Cookie field

can be set to any value.
Normally process responses without a COOKIE OPT option.
Silently ignore responses with more than one COOKIE OPT option.
Silently ignore responses with one COOKIE OPT option if it has an incorrect Resolver Cookie value.
On receipt of a response with one COOKIE OPT option carrying the correct Resolver Cookie value (even if it is a BADCOOKIE error response), the DNS client performs normal response processing, including caching the received Server Cookie, and it MUST change to the Enforced policy for DNS requests to that DNS server IP address. This policy change SHOULD be treated as soft state with the same timeout strategy as the Server Cookie value for that server. On timeout of that state information, the policy for that DNS server reverts to Enabled.

Enforced:

Always include a COOKIE OPT option in requests.
Silently ignore all responses that do not include exactly one COOKIE OPT option having the correct Resolver Cookie value.

[5.2](#) Server Policies and Implementation

A server will logically have one of the following three modes of operation or "policies" for each DNS resolver as distinguished by resolver IP Address and Resolver Cookie.

Disabled:

Ignore COOKIE OPT options in requests.
Never include a COOKIE OPT option in responses.

Enabled:

Always include a COOKIE OPT option in responses.
Normally process requests without a COOKIE OPT option.
Ignore, other than sending a MANYCOOKIE error response, any request with more than one COOKIE OPT option.
Ignore, other than sending a BADCOOKIE error response, any query with one COOKIE OPT option if it has an incorrect Server Cookie.

On receipt of a request with a COOKIE OPT option carrying the correct Server Cookie value, the DNS server performs normal request processing and it SHOULD switch to the Enforced policy for DNS requests from that resolver IP address with that Resolver Cookie in the request. This policy change for that resolver SHOULD be treated as soft state. On timing out that state information, the policy for that resolver IP and Resolver Cookie pair reverts to Enabled.

Enforced:

Always include a COOKIE OPT option in responses.

Ignore requests without a COOKIE OPT option or with more than one COOKIE OPT option, other than returning a NOCOOKIE or MANYCOOKIE error respectively.

Ignore requests with one COOKIE OPT option if they have an incorrect Server Cookie, other than returning a BADCOOKIE error message.

If a request has one COOKIE OPT option with a correct Server Cookie, perform normal processing of the request.

[5.3](#) Implementation Requirements

DNS resolvers and servers SHOULD implement DNS cookies.

DNS resolvers SHOULD operate in and be shipped so as to default to the Enabled or Enforced mode for all servers.

DNS servers SHOULD operate in and be shipped so as to default to the Enabled or Enforced mode for all resolvers they are willing to service.

[6](#). NAT Considerations and AnyCast Server Considerations

In the Classic Internet, DNS Cookies could simply be a pseudo-random function of the resolver IP address and a sever secret or the server IP address and a resolver secret. You would want to compute the Server Cookie that way, so a resolver could cache its Server Cookie for a particular server for an indefinitely amount of time and the server could easily regenerate and check it. You could consider the Resolver Cookie to be a resolver signature over the server IP address which the resolver checks in responses and you could extend this

signature to cover the request ID, for example.

But we have this reality called NAT [[RFC3022](#)], Network Address Translation (including, for the purposes of this document, NAT-PT [[RFC2766](#)], Network Address and Protocol Translation, which has been declared Historic [[RFC4966](#)]). There is no problem with DNS transactions between resolvers and servers behind a NAT box using local IP addresses. Nor is there a problem with NAT translation of internal addresses to external addresses or translations between IPv4 and IPv6 addresses, as long as the address mapping is relatively stable. Should an internal resolver being mapped to a particular external IP address change occasionally, the disruption is no more than when a resolver rolls-over its DNS COOKIE secret. And normally external access to a DNS server behind a NAT box is handled by a fixed mapping which forwards externally received DNS requests to a specific host.

However, NAT devices sometimes also map ports. This can cause multiple DNS requests and responses from multiple internal hosts to be simultaneously mapped to a smaller number of external IP addresses, frequently one. There could be many resolvers behind a NAT box that appear to come from the same source IP address to a server outside that NAT box. If one of these were an attacker (think Zombie or Botnet), that behind-NAT attacker could get the Server Cookie for some server for the outgoing IP address by just making some random request to that server. It could then include that Server Cookie in the COOKIE RR of requests to the server with the forged local IP address of some other host and/or resolver behind the NAT box. (Attacker possession of this Server Cookie will not help in forging responses to cause cache poisoning as such responses are protected by the required Resolver Cookie.)

To fix this potential defect, it is necessary to distinguish different resolvers behind a NAT box from the point of view of the server. It is for this reason that the Server Cookie is specified as a pseudo-random function of both the request source IP address and the Resolver Cookie. From this inclusion of the Resolver Cookie in the calculation of the Server Cookie, it follows that a stable Resolver Cookie, for any particular server, is needed. If, for example, the request ID was included in the calculation of the

Resolver Cookie, it would normally change with each query to a particular server. This would mean that each query would have to be

sent twice: first to learn the new Server Cookie based on this new Resolver Cookie based on the new ID and then again using this new Resolver Cookie to actually get an answer. Thus the input to the Resolver Cookie computation must be limited to the server IP address and one or more things that change slowly such as the resolver secret.

In principle, there could be a similar problem for servers, not particularly due to NAT but due to mechanisms like anycast which may cause queries to a DNS server at an IP address to be delivered to any one of several machines. (External queries to a DNS server behind a NAT box usually occur via port forwarding such that all such queries go to one host.) However, it is impossible to solve this the way the similar problem was solved for NATed resolvers; if the Server Cookie was included in the calculation of the Resolver Cookie the same way the Resolver Cookie is included in the Server Cookie, you would just get an almost infinite series of BADCOOKIE errors as a query was repeatedly retried.

For servers accessed via anycast to successfully support DNS COOKIES, the server clones must either all use the same server secret or the mechanism that distributes queries to them must cause the queries from a particular resolver to go to a particular server for a sufficiently long period of time that extra queries due to changes in Server Cookie resulting from accessing different server machines are not unduly burdensome. When such anycast accessed servers act as recursive servers or otherwise act as resolvers they normally use a different unique address to source their queries and avoid confusion in the delivery of responses.

For simplicity, it is RECOMMENDED that the same server secret be used by each DNS server in a set of anycast servers.

[7](#). Incremental Deployment

The DNS cookies mechanism is designed for incremental deployment and to complement the orthogonal techniques in [[forgery](#)]. Either or both techniques can be deployed independently at each DNS server and resolver.

In particular, a DNS server or resolver that implements the DNS cookies mechanism and is in the Enabled mode will interoperate successfully with a DNS resolver or server that does not implement this mechanism although, of course, in this case it will not get the benefit of the mechanism. When such a server or resolver interoperates with a resolver or server which also implements the DNS cookies mechanism, this is recognized and, for that transaction partner, it enters the Enforced mode and gets the full benefit of the DNS cookies mechanism until this soft state times out and it reverts to Enabled.

[8.](#) IANA Considerations

IANA will allocate the following code points:

The OPT option value for COOKIE is <TBD>.

Three new RCODES are assigned values as listed below:

NOCOKIE is assigned the value ({TBD}, 23 suggested).

BADCOOKIE is assigned the value ({TBD}, 24 suggested).

MANYCOOKIE is assigned the value ({TBD}, 25 suggested).

[9.](#) Security Considerations

DNS Cookies provide a weak form of authentication of DNS requests and responses. In particular, they provide no protection at all against "on-path" adversaries; that is, they provide no protection against any adversary which can observe the plain text DNS traffic, such as an on-path router, bridge, or any device on an on-path shared link (unless the DNS traffic in question on that path is appropriately encrypted).

For example, if a host is connected via an unsecured IEEE 802.11 link (Wi-Fi), any device in the vicinity that could receive and decode the 802.11 transmissions must be considered "on-path". On the other hand, in a similar situation but one where 802.11 Robust Security (WPAv2) is appropriately deployed on the Wi-Fi network nodes, only the Access Point via which the host is connecting is "on-path".

Despite these limitations, use of DNS Cookies on the global Internet is expected to provide a reduction in the available launch points for the traffic amplification and denial of service forgery attacks described in [Section 2](#) above.

The recommended cryptographic algorithms for use in DNS Cookies is HMAC-SHA1-64, that is, the HMAC scheme [[RFC2104](#)] using the SHA1 hash function [[RFC3174](#)] [[RFC4634](#)] with the HMAC output truncated to

64-bits. MD5 is now considered to be susceptible to collisions attacks. Although this does not effect the security of HMAC-MD5, HMAC-SHA1 is believed to be stronger.

In light of the weak plain-text token security provided by DNS Cookies, stronger cryptography is probably not warranted and in many cases it would be acceptable to use the weaker MD5 hash function [[RFC1321](#)]. However, there is nothing wrong with using something stronger, for example, HMAC-SHA256-64 [[RFC4634](#)], assuming a DNS processor has adequate computational resources available. DNS processors that feel the need for somewhat stronger security without a significant increase in computational load should consider more

frequent changes in their resolver and/or server secret; however, this does require more frequent generation of a cryptographically strong random number [[RFC4086](#)] and a change in a server secret will result in a number of initial BADCOOKIE rejected requests from resolvers caching their old Server Cookie.

[10](#). Normative References

[RFC1321] - Rivest, R., "The MD5 Message-Digest Algorithm", [RFC 1321](#), April 1992.

[RFC2104] - Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), February 1997.

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

[RFC2671] - Vixie, P., "Extension Mechanisms for DNS (EDNS0)", August 1999.

[RFC4086] - Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", [BCP 106](#), [RFC 4086](#), June 2005.

[11](#). Informative References

[forgery] - "Measures for making DNS more resilient against forged

answers", Hubert, A. , R. van Mook, [draft-ietf-dnsexp-forgery-resilience-01.txt](#), work in progress, July 2007.

[RFC2766] - Tsirtsis, G., P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", February 2000.

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

[RFC2930] - Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", [RFC 2930](#), September 2000.

[RFC2931] - Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", [RFC 2931](#), September 2000.

[RFC3022] - Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", [RFC 3022](#), January 2001.

[RFC3174] - Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm

1 (SHA1)", [RFC 3174](#), September 2001.

[RFC4033] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), March 2005.

[RFC4034] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", [RFC 4034](#), March 2005.

[RFC4035] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", [RFC 4035](#), March 2005.

[RFC4634] - Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA)", [RFC 4634](#), July 2006.

[RFC4966] - Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", [RFC 4966](#), July 2007.

INTERNET-DRAFT

DNS Cookies

Author's Address

Donald E. Eastlake 3rd
Motorola Laboratories
111 Locke Drive
Marlborough, MA 01752 USA

Telephone: +1-508-786-7554 (w)

E-Mail: Donald.Eastlake@motorola.com

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Additional IPR Provisions

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Expiration and File Name

This draft expires in August 2008.

Its file name is [draft-eastlake-dnsex-03.txt](#)