Network Working Group
Internet-Draft
Expires: October 6, 2004

W. Eddy
NASA GRC/Verizon FNS

April 7, 2004

Mobility Support For TCP
draft-eddy-tcp-mobility-00

Status of this Memo

Copyright Notice

Abstract

Once a TCP connection is established, there are no mechanisms for dynamically rebinding the connection to new IP addresses or port numbers. During the lifetime of a TCP connection, a mobile host's transition between networks (and the corresponding change in its IP address) will break any established TCP connections. As a remedy, this document presents an extension of TCP with support for dynamic bindings of IP addresses and port numbers. Compatibility with existing TCP implementations is maintained, and reasonable steps are taken to prevent remote-redirction attacks.

1. Requirements Notation

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.

2. Introduction

A TCP connection is an association between two pairs of IP address and TCP port number identifiers [1]. None of these four values may change at any point over the lifetime of a TCP connection. Host mobility across networks is a problem that motivates a change to this policy, so that a connection end may be rebound to a new IP addresses and TCP port number without requiring re-establishment. While this feature might also be useful in load balancing scenarios, enabling a busy host to shed some of its connections to another host [2], only the mobility problem is specifically dealt with here.

This document describes a set of TCP options for dynamic rebinding of TCP connection ends. This is accomplished without breaking compatibility with TCP implementations that do not implement these options, and in a manner robust against connection hijacking or remote-redirection attacks, in which some malicious host attempts to change the address bindings of a connection belonging to another host.

For mobility in today's Internet, simply using Mobile IP [3] underneath TCP is not an adequate solution. The Home and Foreign Agents (HAs and FAs) that constitute the Mobile IP infrastructure are not ubiquitously deployed, leaving helpless mobile hosts that either originate in networks lacking an HA or move into networks lacking an FA. Although the need for an FA is mitigated by using colocated care-of addresses in Mobile IP. this document presents a solution that removes the need for both HAs and FAs. Furthermore, the current Mobile IP standard has no provisions for route-optimization, which would remove the less-efficient side of the triangle routes constructed by Mobile IP. Current Internet security practices at many locations actually require the use of Mobile IP's topologically-correct tunnel extension [4], which uses the less efficient indirect route through the HA in both directions. This document describes a solution of which route optimization is an

inherrent property. Stateful firewalls and Network Address translators (NATs) [5] also do not pose problems to the mobility mechanism presented here.

The Mobility Support options for TCP described here allow a host whose IP address has changed to identify itself with a cryptographic cookie and have its TCP connections redirected to the new address. The use of Elliptic Curve Diffie Hellman key exchange [6] to create the cookie secures this technique against remote-redirection attacks.

3. Transport Layer Mobility Architecture

The term "transport layer mobility" is used to describe an architecture for mobility where there is no need for additional network infrastructure beyond traditional IP routers, an automated method of acquiring IP addresses such as the Dynamic Host Configuration Protocol (DHCP) [7] (which is widely deployed), and a means for maintaining host reachability across address changes such as dynamic DNS [8].

A host employing tranport layer mobility may have either a single network interface or multiple interfaces. The host must have some means of knowing when it has moved onto a new IP network. This might be accomplished by receiving a router advertisement from the new network. The mobile host may then use a means such as DHCP to obtain an IP address on the new network. At some point when either the signal strength on the new network is stronger than that on the old network, or a router advertisement on the new network is received, while communications with the old network's router time out, the host should initiate the proceedures this document describes to transition its existing connections to its IP address on the new network, update its reachability bindings via a system such as dynamic DNS, and release its old IP address.

Since there is never any indirection, as through a Mobile IP HA, route optimization is implicit. There are also no concerns about outgoing packets being filtered due to seemingly-spoofed source addresses as in Mobile IP. as only topologically correct addresses

are used with transport layer mobility.

In flight packets sent to the mobile host before its address binding update was received and after it has released its old IP address may be lost and require retransmission. The problem of packets being lost during the handoff between networks exists in Mobile IP as well. We present a way to mitigate the problems this phenomenon causes for transport layer mobility.

While existing connections are maintained, the host's reachability for new connections must be provided by a service like dynamic DNS, as in contrast to Mobile IP, a redirection agent like the HA is not part of the transport layer mobility architecture. Slow convergence of DNS records to new values might pose temporary reachability problems for new connections attempting to reach the mobile host.

The TCP-based transport layer mobility solution described here only allows one of the hosts in a TCP connection to move at a time, and the moving host must either be in the ESTABLISHED or FIN-WAIT states to do so.

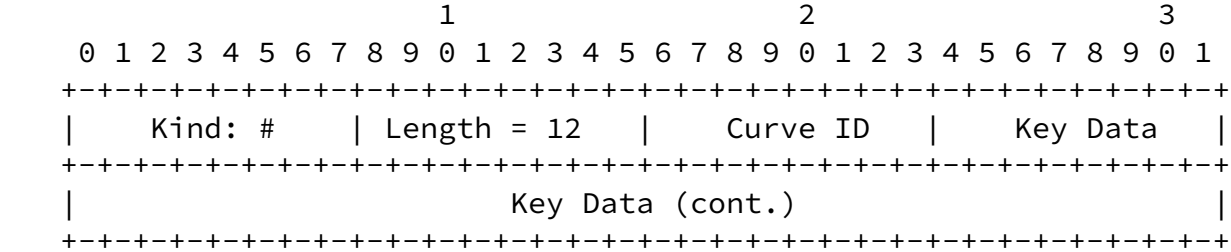
A major advantage of using Mobile IP for mobility is that all transport layer protocols using IP will transparently inherit mobility support without modifications. The mobility method presented in this document applies only to TCP, applications using other transport protocols will not benefit from it. Proposals like Mobile SCTP [9] are needed to give mobility support to other transports. This is somewhat of a replication of effort that simply using Mobile IP avoids. The transparency of Mobile IP handoffs and triangle routes however may cause problems for transport layer protocols that are not designed with the possibility of such occurances in mind. The TCP extension presented here builds on similar works such as the Migrate Internet Mobility Project [10] and TCP-R [11].

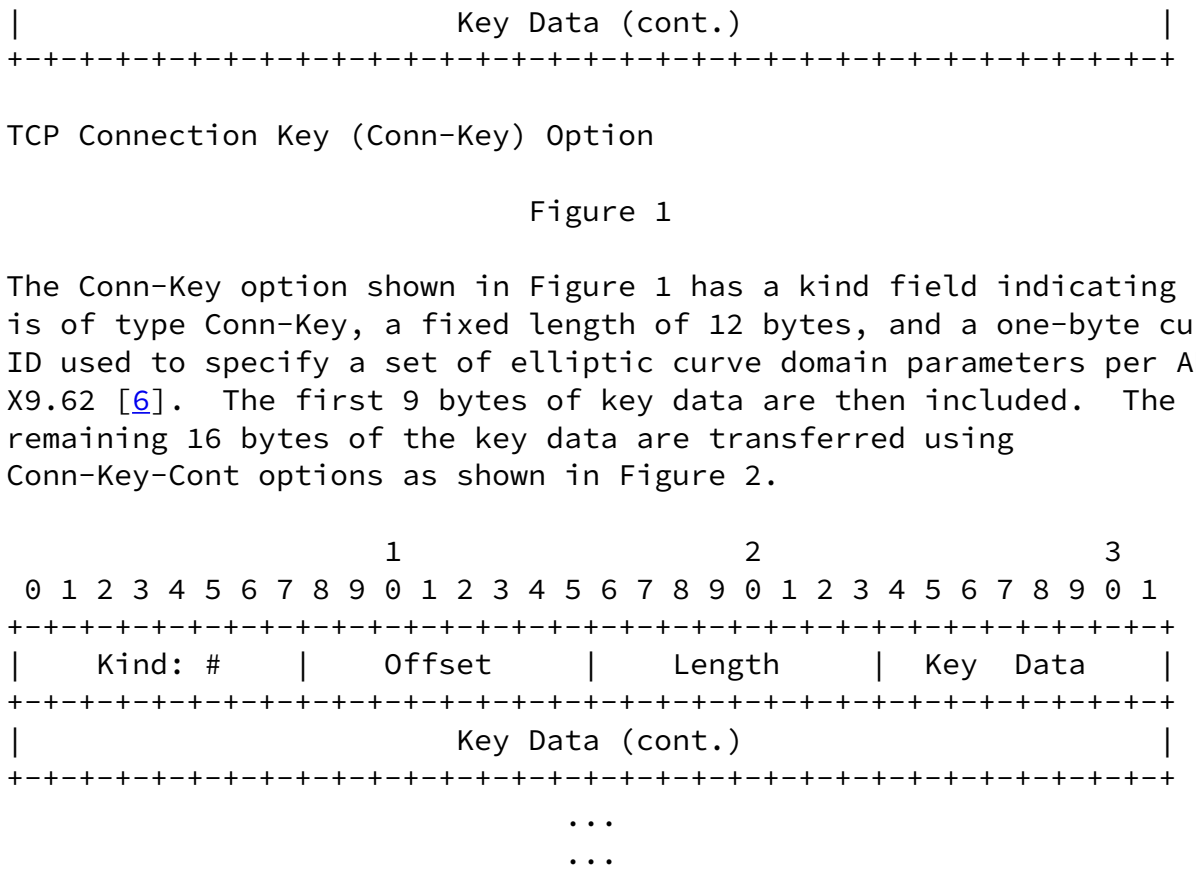
4. Mobility Negotiation

During TCP connection initiation, hosts negotiate whether or not the mobility extension this document presents will be supported for use during the connection. This is accomplished via exchange of the connection key (Conn-Key) option (Figure 1) in the initial SYN and SYN-ACK segments. The Conn-Key option is used to begin the exchange of a 200 bit cryptographic key using the Elliptic Curve Diffie-Hellman (ECDH) method [6]. The Conn-Key option carries the first 9 bytes of the public key data, while the Conn-Key-Cont option (Figure 2) on subsequent segments carries the remaining 16. The public keys are split up over multiple segments in this way because of the tight space limitations on TCP options (see Appendix A). ECDH is used because it generates relatively short and strong keys.

The 200-bit public key is generated by selecting one of the standardized sets of elliptic curve domain parameters (containing a generating point P), and a secret random number X. The key, k, is generated by scalar multiplication over the field specified by the domain parameters via: k = X * P. The encoding of k should be left-padded with zeros to reach 200 bits. The security of the key is baed on the randomness in X's selection. The control block of the connection should store X for use throughout the connection. Hosts that use the SYN cookie technique [12] may not be able to use the mobility methods presented here while they are under attack. Appendix B lists some ways of reducing the overhead of the cryptographic operations used by the mobility options presented in this document.

If a SYN segment initiating a connection does not carry the Conn-Key option, then the SYN-ACK sent in response MUST NOT either, and the mobility extension described in this document MUST be disabled for the connection. Likewise if a host sends a SYN with the Conn-Key option but receives a responding SYN-ACK without the Conn-Key option, the mobility extension MUST be disabled and the Conn-Key-Cont option MUST NOT be used to finish transmitting the key bits.





TCP Connection Key (Conn-Key) Option

Figure 1

The Conn-Key option shown in Figure 1 has a kind field indicating it is of type Conn-Key, a fixed length of 12 bytes, and a one-byte curve ID used to specify a set of elliptic curve domain parameters per ANSI X9.62 [6]. The first 9 bytes of key data are then included. The remaining 16 bytes of the key data are transferred using Conn-Key-Cont options as shown in Figure 2.

TCP Connection Key Continued (Conn-Key-Cont) Option

Figure 2

The offset field in the Conn-Key-Cont option indicates the number of bytes from the beginning of the key data at which the portion of key data contained in this particular option begins. Since 9 bytes of key data are always placed in the Conn-Key option, offset should always be greater than eight. The Conn-Key-Cont options may be of variable length to accomodate room for other TCP options. Ideally, the 19 bytes required to transfer the entire remaining portion of the key data would be available in the first segment after the one carrying the Conn-Key option. In this case only a single Conn-Key-Cont option need be sent. Otherwise, the remaining data can be broken up amongst multiple Conn-Key-Cont options. If any segments carrying either Conn-Key or Conn-Key-Cont options are retransmitted, the retransmissions MUST include identical options to the original

transmissions. If a host receives inconsistent overlapping key data, it MUST reset the connection.

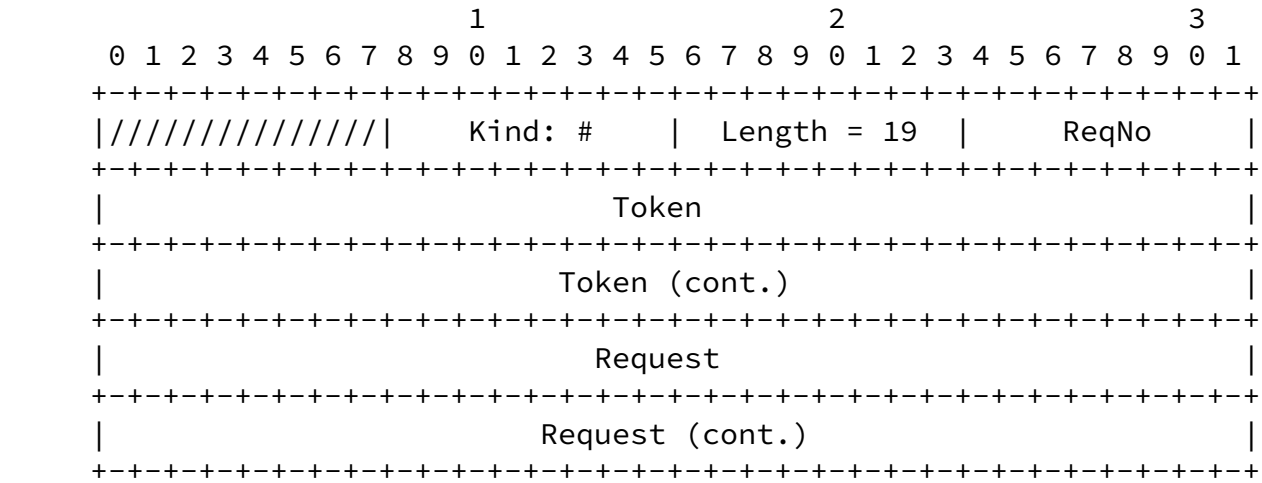
After 200 bits of key data, k, have been received, a host can generate the connection's shared secret key, K, via the scalar multiplication over the selected field: K = k * X. When both sides have finished the exchange, a shared secret cryptographic key is then available for use in any number of TCP extensions. For transport layer mobility, it is used to authenticate that a location update indeed comes from the host that originally made the connection and is not an attempted remote redirection attack. Potentially, the exchanged key might be useful in other domains as well.

5. Location Updates

Once the full exchange of key data is successfully completed, either host may change IP addresses freely without breaking the connection, so long as they do not both do so simultaneously.

To minimize any problems with segments and acknowledgements being sent by the remote host in time between when the old address is fully released and the new one aquired, a host MAY anticipatorily pause the transmission of any data segments its buffers hold and send an advertised receive window of zero to pause the other host. This technique could significantly reduce the possibility of lost segments during the handover.

After a host changes addresses, it resets its congestion control state to the intial slow-start conditions, [13], and resets its estimates of RTT, RTTVAR, and RTO [14]. The mobile host then sends a TCP segment with the SYN bit set and the Location Update (Loc-Update) option. The SYN bit is set in order to allow the segment to pass through NATs and stateful firewalls that will see the segment as invalid because they haven't seen a SYN seeming to initiate the connection yet. The sequence number of the segment should be that of the last acknowledgement received. Any data transmitted above the last acknowledgement before the address change will require retransmission if not covered by SACK blocks. The acknowledgement field in the SYN should indicate the last byte of data received. The format of the Loc-Update option is shown in Figure 3.



TCP Location Update (Loc-Update) Option

Figure 3

Loc-Update options are fixed at 19 bytes. They contain a request number (ReqNo), a token used to lookup the connection to be updated,

and a request that authenticates it. The token (T) consists of the upper 64 bits of the SHA-1 hash [15] over the initial sequence number of the original SYN segment that begun the connection (N_i), the sequence number in the corresponding SYN-ACK (N_a), and the shared secret key (K). It can be computed via: T = upper64(SHA-1(N_i, N_a, K)), where upper64() selects the 64 most significant bits in the 160-bit value computed by the hash function SHA-1(). If observed by a malicious host, this value is replayable, although since we do not use it for authentication, but only for identification of a candidate mobile connection, this is not a problem. The request (R) is computed in a similar manner, except with the hash also covering the sequence number of the SYN (S), and the request number, such that: R = upper64(SHA-1(N_i, N_a, K, S, ReqNo)). The initial request number may be chosen by the host sending the option in any manner desired, so long as it is incremented by one every time that a location update is sent. The request number rolls back to zero after reaching its maximum value.

Upon receiving a SYN carrying a location update, a host first attempts to look up the token it contains. If the token does not belong to any known connections, the segment is immediately rejected and no response is sent. If the token does match a known connection, then the request can be validated by computing the hash and comparing it to the received value. If the values match, then the host resets its state in the same manner specified for the sender of the Loc-Update option, and the remote host's address and port number are updated in the control block. The SYN should be acknowledged with a SYN-ACK carrying the sequence number of the acknowledgment field in the SYN segment. Data above this point will require retransmission if not covered by a SACK block. Updating the remote port number normally shouldn't be necessary, however it might be required in case the new IP address is behind a NAT, and it doesn't cause any additional difficult, so we do it by default.

On receiving the SYN-ACK, the host that sent the location update resumes normal operation. Either side may change addresses again at

this point.

6. State Machine Changes

There is a possibility of connections from being accidentally closed by stray segments sent during a mobile host's handover between networks. To prevent this, when the mobility options are used on a connection, then a transition is added between the ESTABLISHED state and TIME-WAIT on the reception of a RST. and the TIME-WAIT state is slightly modified to contain a transition back to ESTABLISHED if a valid request in a Loc-Update option is received before the timeout, These transitions need not be added for connections that do not use the mobility options defined in this document. To motivate these changes, we provide an example.

Consider the scenario where a mobile host, MH, is temporarily bound to address A and has an active connection with a corresponding host CH. MH moves off of the network A belongs to, and before it is able to get a new address, A is reaped and reassigned to another host. Since CH hasn't seen a location update from MH, it still sends segments addressed to A. The new host now bound to A has no knowledge of the connection and replies with a RST. This would normally cause CH to immediately close the connection. CH should, however, instead take the RST as meaning that the MH has possibly moved and go into the TIME-WAIT state. The transition to the modified TIME-WAIT state SHOULD occur if the mobility extensions described in this document are enabled.

7. Security Considerations

The authentication mechanism used for location updates is based on the initial ECDH key exchange and a SHA-1 hash. Any presently unknown vulnerabilities in ECDH or SHA-1 may make the source of location updates less certain and open a connection with these extensions up to remote redirection attacks.

The TCP extensions described in this document contribute additional state per connection. This makes hosts implementing them more vulnerable to SYN flood attacks. Some of the implementation techniques in Appendix B may be helpful in reducing the impact these extensions cause.

8 References

[1] Postel, J., "Transmission Control Protocol", RFC 793, September 1981.

[2] Snoeren, A., Andersen, D. and H. Balakrishnan, "Fine-Grained Failover Using Connection Migration", Proc. of the Third Annual USENIX Symposium on Internet Technologies and Systems (USITS),

March 2001.

[3] Perkins, C., "IP Mobility Support for IPv4", RFC 3220, January 2002.

[4] Montenegro, G., "Reverse Tunneling for Mobile IP, Revised", RFC 3024, January 2001.

[5] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[6] American National Standards Institute, "Public Key Cryptography for the Financial Security Industry", ANSI X9.62, January 1999.

[7] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

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

[9] Koh, S., Lee, M., Riegel, M., Ma, M. and M. Tuexen, "Dynamic Host Configuration Protocol", draft-sjkok-sctp-mobility-03 (work in progress), February 2004.

[10] Snoeren, A. and H. Balakrishnan, "An End-to-End Approach to

Host Mobility", Proc. of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, August 2000.

[11] Funato, D., Yasuda, K. and H. Tokuda, "TCP-R: TCP Mobility Support for Continuous Operation", International Conference on Network Protocols (ICNP), October 1997.

[12] Bernstein, D., "SYN cookies", September 1996.

[13] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[14] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[15] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[16] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[17] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996.

Author's Address

Wesley M. Eddy
NASA GRC/Verizon FNS

EMail: weddy@grc.nasa.gov

Appendix A. Differences From the Migrate-Permitted Option

While based on Snoeren et al's Migrate work [10], the exchange of key data described in this document has two main differences, (1) we leave some space open in the SYN and SYN-ACK's TCP headers for additional options, and (2) we do not overload the meaning of data in the time-stamp and time-stamp echo fields on those packets.

The maximum number of bytes that can be used for TCP options is 40. This limitation comes from the 4-bit length of the Data Offset field of the TCP header [1]. The maximum value this field can encode is 15, which represents the number of 4-byte quantities in the TCP header. Thus, the maximum TCP header is 60 bytes. Since the fixed length fields take up 20 bytes, this leaves 40 bytes for options.

In addition to the Conn-Key option, there are several other TCP options which must be present in the initial SYN and SYN-ACK segments in order for their features to be used on a connection. At this time, the defined TCP options sharing this property are MSS [1], window scale [16], timestamp [16], and selective acknowledgements permitted [17]. Based on the lengths of these previously defined options alone (MSS = 4 bytes, window scale = 3 bytes, timestamp = 10 bytes, and selective acknowledgements permitted = 2 bytes, total = 19 bytes), there are only 21 bytes available for other options that must go into the SYN packets. The originally-proposed Migrate-Permitted Option [10] used 20 of these, leaving only a single byte free, which is not enough for even a single other option, given the minium option length of two bytes (one for kind and one for length).

There is a resulting inability (or at least extreme difficulty) involved in defining future TCP options compatible with the simultaneous use of the Migrate-Permitted Option and previous options. To mitigate this, we have spread the key data originally exchanged via this option out over two segments with the Conn-Key and Conn-Key-Cont options. A downside to this is that an extra RTT is imposed upon the exchange of key data, meaning a mobile host cannot change addresses for two full RTTs after it initiates a connection by sending a SYN, and one RTT after receiving a SYN and sending a SYN-ACK. This does not seem like an unreasonable restriction. The benefit obtained by this key data encoding is that it leaves some room for new TCP options that may require negotiation at connection startup.

The original Migrate-Permitted Option also required the presence of the timestamp option and used 8 of the timestamp option's bytes to encode key data. This was a reasonable approach since the timestamp data in the SYN and SYN-ACK segments was not to be used (at least for RTTM and PAWS) anyways [16]. However, this approach does not leave

open the possibility that later TCP extensions may have better use

for these fields, perhaps even a use more closely aligned with their purpose as timestamps than as key data. By removing the 8 bytes of key data encoded as timestamps by the Migrate-Permitted option and moving them into the Conn-Key option, we impose the transmission of an additional 8 bytes per direction upon startup of a TCP connection. This should have little noticable effect on TCP performance.

Appendix B. Overhead Reduction

The overhead of running cryptographic routines is typically much greater than normal for common TCP implementations. We describe some possible approaches to minimizing this overhead.

An easy means of preventing the overhead of these options from being an issues is to only enable the mobility options by an applications request. In this way, typically short-lived applications like name-lookup and remote procedure calls may elect not to have the mobility options enabled. Such short-lived connections are the ones where the additional cryptographic routine latency would be most problematic and that would benefit from mobility the least, as the potential for movement across networks will be small over the short lifetimes of connections. Typically more long-lived applications like file transfer or remote shells would, in contrast, be less adversely affected by a small additional amount of startup delay and would be more prone to mobility-induced disconnection during the connections lifetime, and so such flows could select to enable the options described in this document.

The security of the ECDH key exchange depends on selection of a good random number X. Presently, generation of good random numbers is a particularly time-consuming operation. To lessen the latency that on-demand generation of X values would impose, it should be sufficient to pre-generate a list of X values to be used for connections in the near future. A single value of X might also be shared by a small number of connections, although this approach should only be taken if absolutely required, as it provides more information about X to eavesdropping hosts.

The validation tokens used to identify that a SYN carrying a location update refers to a particular connection SHOULD be precomputed and are stable over the lifetime of a connection. An efficient data structure for looking these up MAY be implemented.

Intellectual Property Statement

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 IETF's procedures with respect to rights in IETF 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.

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.