

Network Working Group  
Request for Comments: 5327  
Category: Experimental

S. Farrell  
Trinity College Dublin  
M. Ramadas  
ISTRAC, ISRO  
S. Burleigh  
NASA/Jet Propulsion Laboratory  
September 2008

## **Licklider Transmission Protocol - Security Extensions**

### Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

### IESG Note

This RFC is not a candidate for any level of Internet Standard. It represents the consensus of the Delay Tolerant Networking (DTN) Research Group of the Internet Research Task Force (IRTF). It may be considered for standardization by the IETF in the future, but the IETF disclaims any knowledge of the fitness of this RFC for any purpose and in particular notes that the decision to publish is not based on IETF review for such things as security, congestion control, or inappropriate interaction with deployed protocols. See [RFC 3932](#) for more information.

### Abstract

The Licklider Transmission Protocol (LTP) is intended to serve as a reliable convergence layer over single-hop deep-space radio frequency (RF) links. LTP does Automatic Repeat reQuest (ARQ) of data transmissions by soliciting selective-acknowledgment reception reports. It is stateful and has no negotiation or handshakes. This document describes security extensions to LTP, and is part of a series of related documents describing LTP.

This document is a product of the Delay Tolerant Networking Research Group and has been reviewed by that group. No objections to its publication as an RFC were raised.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction</a>	<a href="#">2</a>
<a href="#">2.</a>	<a href="#">Security Extensions</a>	<a href="#">2</a>
<a href="#">2.1.</a>	<a href="#">LTP Authentication</a>	<a href="#">3</a>
<a href="#">2.2.</a>	<a href="#">A Cookie Mechanism</a>	<a href="#">6</a>
<a href="#">3.</a>	<a href="#">Security Considerations</a>	<a href="#">7</a>
<a href="#">4.</a>	<a href="#">IANA Considerations</a>	<a href="#">7</a>
<a href="#">5.</a>	<a href="#">Acknowledgments</a>	<a href="#">8</a>
<a href="#">6.</a>	<a href="#">References</a>	<a href="#">8</a>
<a href="#">6.1.</a>	<a href="#">Normative References</a>	<a href="#">8</a>
<a href="#">6.2.</a>	<a href="#">Informative References</a>	<a href="#">9</a>

## [1.](#) Introduction

This document describes extensions to the base LTP protocol [[LTPSPEC](#)]. The background to LTP is described in the "motivation" document [[LTPMOTIVE](#)]. All the extensions defined in this document provide additional security features for LTP.

LTP is designed to provide retransmission-based reliability over links characterized by extremely long message round-trip times (RTTs) and/or frequent interruptions in connectivity. Since communication across interplanetary space is the most prominent example of this sort of environment, LTP is principally aimed at supporting "long-haul" reliable transmission in interplanetary space, but has applications in other environments as well.

This document describes security extensions to LTP, and is part of a series of related documents describing LTP. Other documents in this series cover the motivation for LTP and the main protocol specification. We recommend reading all the documents in the series before writing code based on 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 [[B97](#)].

## [2.](#) Security Extensions

The syntactical layout of the extensions are defined in [Section 3.1.4](#) of the base protocol specification [[LTPSPEC](#)].

Implementers should note that the LTP extension mechanism allows for multiple occurrences of any extension tag, in both (or either) the header or trailer. For example, the LTP authentication mechanism defined below requires both header and trailer extensions, which both use the same tag.



This document defines new security extensions for LTP but does not address key management since key management in Delay-Tolerant Networking (DTN) remains an open research question.

If LTP were deployed layered on top of UDP, it might be possible to use IPsec or other existing security mechanisms. However, in general DTN, IPsec's key exchange (IKE) cannot work (e.g., where link delays are measured in minutes).

### **2.1. LTP Authentication**

The LTP authentication mechanism provides cryptographic authentication of the segment.

Implementations MAY support this extension field. If they do not support this header, then they MUST ignore it.

The LTP authentication extension field has the extension tag value 0x00.

LTP authentication requires three new fields, the first two of which are carried as the value of the Extensions field of the LTP segment header, and the third of which is carried in the segment trailer.

The fields that are carried in the header extensions field are catenated together to form the extension value (with the leftmost octet representing the ciphersuite and the remaining octets the KeyID). The KeyID field is optional, and is determined to be absent if the extension value consists of a single octet.

Ciphersuite: an 8-bit integer value with values defined below.

KeyID: An optional key identifier, the interpretation of which is out of scope for this specification (that is, implementers MUST treat these KeyID fields as raw octets, even if they contained an ASN.1 DER encoding of an X.509 IssuerSerial construct [[PKIXPROF](#)], for example).

The LTP-auth header extension MUST be present in the first segment from any LTP session that uses LTP authentication, but MAY be omitted from subsequent segments in that session. To guard against additional problems arising from lost segments, implementations SHOULD, where bandwidth allows, include these fields in a number of segments in the LTP session. If the first segment (or any part thereof) is retransmitted, then the LTP-auth header extension MUST be included in the retransmission.



The field carried as a trailer extension is the AuthVal field. It contains the authentication value, which is either a message authentication code (MAC) or a digital signature. This is itself a structured field whose length and formatting depend on the ciphersuite.

If for some reason the sender includes two instances of LTP-auth headers, then there is a potential problem for the receiver in that presumably at least one of the AuthVal fields will not verify. There are very few situations where it would make sense to include more than one LTP-auth extension in a single segment, since LTP is a peer-to-peer protocol. If however, keys are being upgraded, then the sender might protect the segment with both the new and old keys. In such cases, the receiver MUST search and can consider the LTP authentication valid so long as one AuthVal is correct.

For all ciphersuites, the input to the calculation is the entire encoded segment including the AuthVal extension tag and length, but not of course, including the AuthVal value.

We define three ciphersuites in this specification. Our approach is to follow the precedent set by TLS [TLS], and to "hardcode" all algorithm options in a single ciphersuite number. This means that there are 256 potential ciphersuites supported by this version of LTP-auth. Since this is a limited space, IANA has established a registry for LTP Ciphersuites as described in the IANA Considerations section below. Current ciphersuite assignments are:

Ciphersuite	Value
-----	-----
HMAC-SHA1-80	0
RSA-SHA256	1
Unassigned	2-127
Reserved	128-191
Private/Experimental Use	192-254
NULL	255

#### 1. HMAC-SHA1-80 Ciphersuite

The HMAC-SHA1-80 ciphersuite involves generating a MAC over the LTP segment and appending the resulting AuthVal field to the end of the segment. There is only one MACing algorithm defined for this, which is HMAC-SHA1-80 [HMAC]. The AuthVal field in this case contains just the output of the HMAC-SHA1-80 algorithm, which is a fixed-width field (10 octets).



## 2. RSA-SHA256 Ciphersuite

The RSA-SHA256 ciphersuite involves generating a digital signature of the LTP segment and appending the resulting AuthVal field to the end of the segment. There is only one signature algorithm currently defined for this, which is RSA with SHA256 as defined in [RSA], Section 8.2. The AuthVal field in this case is simply the signature value, where the signature value occupies the minimum number of octets, e.g., 128 octets for a 1024-bit signature).

## 3. NULL Ciphersuite

The NULL ciphersuite is basically the same as the HMAC-SHA1-80 ciphersuite, but with a hardcoded key. This ciphersuite effectively provides only a strong checksum without authentication, and thus is subject to active attacks and is the equivalent of providing a Cyclic Redundancy Check (CRC).

The hardcoded key to be used with this ciphersuite is the following:

```
HMAC_KEY      : c37b7e64 92584340
                : bed12207 80894115
                : 5068f738
```

(The above is the test vector from [RFC 3537](#) [WRAP].)

In each case, the bytes that are input to the cryptographic algorithm consist of the entire LTP segment except the AuthVal. In particular, the header extensions field that may contain the ciphersuite number and the KeyID field is part of the input.

The output bytes of the cryptographic operation are the payload of the AuthVal field.

The following shows an example LTP-auth header, starting from and including the Extensions field.

```
ext tag sdnv c-s k-id
+---+---+---+---+---+
|0x11|0x00|0x02|0x00|0x24|
+---+---+---+---+---+
```





The Extensions field has the value 0x11 with the most significant and least significant nibble value 1, indicating the presence of one header and one trailer extension, respectively. The next octet is the extension tag (0x00 for LTP-auth), followed by the Self-Delimiting Numeric Value (SDNV) encoded length of the ensuing data: a one-octet ciphersuite (0x00 meaning HMAC-SHA1-80) and the KeyID (in this case with a short value of 0x24). The trailer extension (not shown above) should contain the AuthVal.

## **2.2. A Cookie Mechanism**

The use of cookies is a well-known way to make Denial of Service (DoS) attacks harder to mount. We define the cookie extension for use in environments where an LTP implementation is liable to such attacks.

The cookie is placed in a header extension field, and has no related trailer extension field. It has the extension tag value 0x01.

The cookie value can essentially be viewed as a sufficiently long random number, where the length can be determined by the implementation (longer cookies are harder to guess and therefore better, though using more bandwidth). Note that cookie values can be derived using lots of different schemes so long as they produce random-looking and hard-to-predict values.

The first cookie inserted into a segment for this session is called the initial cookie.

Note that cookies do not outlast an LTP session.

The basic mode of operation is that an LTP engine can include a cookie in a segment at any time. After that time, all segments corresponding to that LTP session MUST contain a good cookie value -- that is, all segments both to and from the engine MUST contain a good cookie. Clearly, there will be some delay before the cookie is seen in incoming segments -- implementations MUST determine an acceptable delay for these cases, and MUST only accept segments without a cookie until that time.

The cookie value can be extended at any time by catenating more random bits. This allows both LTP engines to contribute to the randomness of the cookie, where that is useful. It also allows a node that considers the cookie value too short (say due to changing circumstances) to add additional security. In this case, the extended cookie value becomes the "to-be-checked-against" cookie value for all future segments (modulo the communications delay as above).



It can happen that both sides emit segments containing an initial cookie before their peer has a chance to see any cookie. In that case, two cookie extension fields **MUST** be included in all segments subsequently (once the traffic has caught up). That is, the sender and recipient cookies are handled independently. In such cases, both cookie values **MUST** be "good" at all relevant times (i.e., modulo the delay). In this case, the peer's initial cookie **MUST** arrive before the calculated delay for receipt of segments containing this engine's cookie -- there is only a finite window during which a second cookie can be inserted into the session.

A "good" cookie is therefore one that starts with the currently stored cookie value, or else a new cookie where none has been seen in that session so far. Once a cookie value is seen and treated as "good" (e.g., an extended value), the previous value is no longer "good".

Modulo the communications delay, segments with an incorrect or missing cookie value **MUST** be silently discarded.

If a segment is to be retransmitted (e.g., as a result of a timer expiring), then it needs to contain the correct cookie value at the time of (re)transmission. Note that this may differ from what was the correct cookie value at the time of the original transmission.

### **3. Security Considerations**

The extensions specified above are generally intended to help thwart DoS attacks. For environments where lower layers provide neither integrity nor freshness, it makes sense to use both extensions together. For example, in the case where a node extends an existing cookie, the lack of origin authentication would allow a man in the middle to lock out the session.

While there are currently some concerns about using the SHA-1 algorithm, these appear to only make it easier to find collisions. In that case, the use of HMAC with SHA-1 can still be considered safe. However, we have changed to use SHA-256 for the signature ciphersuite.

### **4. IANA Considerations**

IANA has created and now maintains registry for known LTP ciphersuites (as defined in [Section 2.1](#)). The registry has been populated using the initial values given in Sections [2.1](#) and [2.2](#) above. IANA may assign LTP Extension Tag values from the range 2..127 (decimal, inclusive) using the Specification Required rule [[GUIDE](#)]. The specification concerned can be an RFC (whether



Standards Track, Experimental, or Informational), or a specification from any other standards development organization recognized by IANA or with a liaison with the IESG, specifically including CCSDS (<http://www.ccsds.org/>).

## 5. Acknowledgments

Many thanks to Tim Ray, Vint Cerf, Bob Durst, Kevin Fall, Adrian Hooke, Keith Scott, Leigh Torgerson, Eric Travis, and Howie Weiss for their thoughts on this protocol and its role in Delay-Tolerant Networking architecture.

Part of the research described in this document was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work was performed under DOD Contract DAA-B07-00-CC201, DARPA AO H912; JPL Task Plan No. 80-5045, DARPA AO H870; and NASA Contract NAS7-1407.

Thanks are also due to Shawn Ostermann, Hans Kruse, and Dovel Myers at Ohio University for their suggestions and advice in making various design decisions. This work was done when Manikantan Ramadas was a graduate student at the EECS Dept., Ohio University, in the Internetworking Research Group Laboratory.

Part of this work was carried out at Trinity College Dublin as part of the Dev-SeNDT contract funded by Enterprise Ireland's technology development programme.

## 6. References

### 6.1. Normative References

- [B97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [GUIDE] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), February 1997.
- [LTPSPEC] Ramadas, M., Burleigh, S., and S. Farrell, "Licklider Transmission Protocol - Specification", [RFC 5326](#), September 2008.



[RSA] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", [RFC 3447](#), February 2003.

## **6.2. Informative References**

[LTPMOTIVE] Burleigh, S., Ramadas, M., and S. Farrell, "Licklider Transmission Protocol - Motivation", [RFC 5325](#), September 2008.

[PKIXPROF] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), May 2008.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.

[WRAP] Schaad, J. and R. Housley, "Wrapping a Hashed Message Authentication Code (HMAC) key with a Triple-Data Encryption Standard (DES) Key or an Advanced Encryption Standard (AES) Key", [RFC 3537](#), May 2003.





## Authors' Addresses

Stephen Farrell  
Computer Science Department  
Trinity College Dublin  
Ireland  
Telephone: +353-1-896-1761  
EMail: [stephen.farrell@cs.tcd.ie](mailto:stephen.farrell@cs.tcd.ie)

Manikantan Ramadas  
ISRO Telemetry Tracking and Command Network (ISTRAC)  
Indian Space Research Organization (ISRO)  
Plot # 12 & 13, 3rd Main, 2nd Phase  
Peenya Industrial Area  
Bangalore 560097  
India  
Telephone: +91 80 2364 2602  
EMail: [mramadas@gmail.com](mailto:mramadas@gmail.com)

Scott C. Burleigh  
Jet Propulsion Laboratory  
4800 Oak Grove Drive  
M/S: 301-485B  
Pasadena, CA 91109-8099  
Telephone: +1 (818) 393-3353  
Fax: +1 (818) 354-1075  
EMail: [Scott.Burleigh@jpl.nasa.gov](mailto:Scott.Burleigh@jpl.nasa.gov)



## Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#) and at <http://www.rfc-editor.org/copyright.html>, 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.

## Intellectual Property

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](mailto:ietf-ipr@ietf.org).

