

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 30, 2016

D. Migault  
K. Ma  
Ericsson  
R. Rich  
Akamai  
S. Mishra  
Verizon Communications  
O. Gonzales de Dios  
Telefonica  
June 28, 2016

LURK TLS/DTLS Use Cases  
draft-mglt-lurk-tls-use-cases-02

## Abstract

TLS has been designed to setup and authenticate transport layer between a TLS Client and a TLS Server. In most cases, the TLS Server both terminates the TLS Connection and owns the authentication credentials necessary to authenticate the TLS Connection.

This document provides use cases where these two functions are split into different entities, i.e. the TLS Connection is terminated on an Edge Server, while authentication credentials are generated by a Key Server, that owns the Private Key.

## 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 December 30, 2016.

Internet-Draft

LURK/TLS Use Cases

June 2016

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

<a href="#">1.</a>	Requirements notation . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">4.</a>	Architecture Overview . . . . .	<a href="#">4</a>
<a href="#">5.</a>	Use cases . . . . .	<a href="#">5</a>
<a href="#">5.1.</a>	Containers and Virtual Machines Use Cases . . . . .	<a href="#">5</a>
<a href="#">5.1.1.</a>	Description . . . . .	<a href="#">5</a>
<a href="#">5.1.2.</a>	LURK Expectations . . . . .	<a href="#">6</a>
<a href="#">5.2.</a>	Content Provider Use Case . . . . .	<a href="#">7</a>
<a href="#">5.2.1.</a>	Description . . . . .	<a href="#">7</a>
<a href="#">5.2.2.</a>	LURK Expectations . . . . .	<a href="#">7</a>
<a href="#">5.3.</a>	Content Owner / Content Provider Use Case . . . . .	<a href="#">8</a>
<a href="#">5.3.1.</a>	Description . . . . .	<a href="#">8</a>
<a href="#">5.3.2.</a>	LURK Expectations . . . . .	<a href="#">8</a>
5.4.	Content Distribution Network Interconnection Use Case .	9
<a href="#">5.4.1.</a>	Description . . . . .	<a href="#">9</a>
<a href="#">5.4.2.</a>	LURK Expectations . . . . .	<a href="#">9</a>
<a href="#">6.</a>	Requirements . . . . .	<a href="#">10</a>
<a href="#">6.1.</a>	LURK Requirements . . . . .	<a href="#">10</a>
<a href="#">6.2.</a>	Key Server Requirements . . . . .	<a href="#">10</a>
<a href="#">6.3.</a>	Edge Server Requirements . . . . .	<a href="#">11</a>
<a href="#">7.</a>	Security Considerations . . . . .	<a href="#">11</a>
<a href="#">8.</a>	IANA Considerations . . . . .	<a href="#">11</a>
<a href="#">9.</a>	Acknowledgements . . . . .	<a href="#">11</a>
<a href="#">10.</a>	References . . . . .	<a href="#">11</a>
<a href="#">10.1.</a>	Normative References . . . . .	<a href="#">11</a>

<a href="#">10.2.</a>	Informative References . . . . .	<a href="#">12</a>
	Authors' Addresses . . . . .	<a href="#">12</a>

## [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 [[RFC2119](#)].

## [2.](#) Introduction

TLS has been designed for end-to-end security between a TLS Server and a TLS Client. As TLS is widely used to provide an authenticated channel between applications, the following models assumes that applications end points and connectivity end point are combined. In that case, authentication of the connection end point and authentication of the application end point could be combined and assimilated as a single authentication.

Such assumption for the TLS model may not be true especially in the current web architecture where application content is not necessarily associated with the connection end point. For example, Content Delivery Network are in charge of delivering content that they may or may are not own.

This document provide use case where authentication of the TLS Server involves multiple parties or entities as opposed to a single entity in the standard TLS model.

## [3.](#) Terminology

**TLS Client:** The TLS Client designates the initiator of the TLS session. The terminology is the one of [[RFC5246](#)]. The current document considers that the TLS Client and the application initiating the session are hosted on the same host. If not they are hosted on the same administrative domain with a trust relation between the TLS Client and the application. In other words, the client endpoint is considered to be a single entity as described initially in [[RFC5246](#)].

**TLS Server:** The TLS Server designates the endpoint of a TLS session initiated by the TLS Client. In the standard TLS, the TLS Server, is both the terminating point of the TLS Connection and the entity authenticated by the TLS Client. This document considers that the TLS Server be split into the Edge Server terminating the TLS Connection and the Key Server providing the necessary capabilities to the TLS Client to proceed with the authentication.

**Private Key:** is the cryptographic credential used by the TLD client to authenticate the TLS Server. The purpose of this document

is to enable the Private Key to be hosted in the Key Server, that is, outside of the TLS Connection terminating point.

**TLS Connection:** The authenticated TLS Connection between the TLS Client and the TLS Server. In this document, the TLS connection terminates on the Edge Server and enables authenticating the Private Key hosted on the Key Server.

**Key Server:** The server hosting the Private Key. The Key Server provides an interface that enable cryptographic operations to be performed remotely by the Edge Servers.

**Edge Server:** The Edge Server designates a node that handles traffic for a Content Provider. A TLS client initiates a TLS session to authenticate a Content provider, but may in fact be served by an Edge Server likely belonging to a different administrative domain.

**Content Owner:** The owner of the content. This is the entity requested by the application of the TLS Client.

**Content Provider:** The entity responsible to deliver the content. In some cases, the Content Provider also own its own Content Delivery Network.

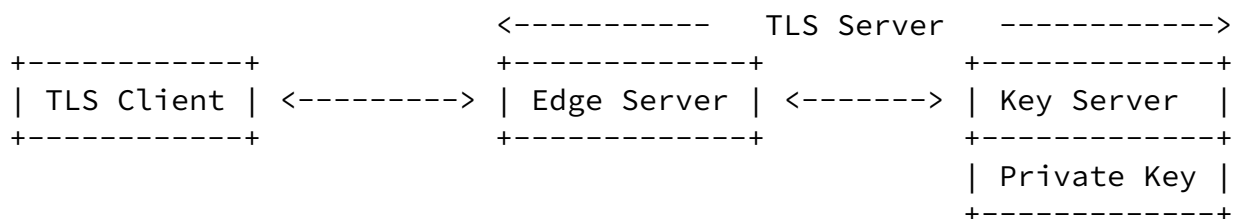
**Content Delivery Network (CDN):** designates a organization in charge of managing delivery of a content on behalf of a Content Provider. In most cases, the CDN is a different organization than the Content Provider.

#### 4. Architecture Overview

Figure 1 provides an overview of the architecture considered by the different uses cases exposed in this document.

The TLS Client initiates a TLS connection with the Edge Server (1) which is expected to be authenticated by its Private Key. The Edge Server terminates the TLS connection of the TLS Client, but does not own the Private Key. Instead, the Private Key is owned by the Key Server, which performs all cryptographic operations on behalf of the Edge Server. Upon request from the Edge Server, the Key Server provides the authentication credentials to the Edge Server (2). These authentication credentials depend on the authentication methods agreed between the TLS Client and the Edge Server as well as the capabilities of the Key Server. The Authentication Credentials returned by the Key Server enables the Edge Server to complete the TLS Handshake and the TLS Client to authenticate the Edge Server as a key owner of the Private Key (3).

The above described architecture to split the standard TLS Server into two functions: the Edge Server that terminates the TLS Connection and the Key Server to host the Private Key and perform all necessary cryptographic operations for the authentication.



##### 1. TLS Connection Initialization

----->

##### 2. Authentication Credentials (Private Key based cryptographic operations)

<----->

##### 3. Finalization of the TLS Connection Handshake

<----->

TLS Connection Established  
<=====>

Figure 1: TLS Session Key Interface Architecture

## [5.](#) Use cases

### [5.1.](#) Containers and Virtual Machines Use Cases

#### [5.1.1.](#) Description

In virtual environment application servers may run within virtual machines or containers. When TLS is used to provide an authenticated and encrypted communication channel to the application, it is currently common that the container or the virtual machine hosts the Private Key used for the authentication. Hosting multiple copies of the Private Key through the cloud increases the risk of leaking the Private Key.

For example, virtualization and persistent storage of virtual machines or containers image over different places in the cloud may result in multiple copies of the Private Key through the cloud. In addition, operating system level virtualization is a virtualization method with a very low overhead. Isolation is performed at the process and kernel level, which may provide a smaller isolation compared to the one provided by the traditional hypervisors. With

lighter isolation, containers avoid storing private and highly sensitive data such as a Private Key.

#### [5.1.2.](#) LURK Expectations

In this scenario, LURK enables the cloud provider to store the Private Key in a centralized Key Server that is remotely accessed by the different instances of the virtual machines or containers. As containers or virtual machines are not hosting the Private Key, the risks of leakage due to a lack of strong isolation or due to a replication of the image is reduced. On the other hand, each container or virtual machine is likely to interact with the Key Server in an authenticated way, in which case, credentials used for the authentication is exposed to risks of leakage that are similar to those the Private Key were exposed without LURK. As a result, LURK

does not reduce the risk of leakage itself, but transfers the risk from the Private Key to the containers or virtual machines' authentication credentials. In fact, the consequence of a leakage of authentication credentials is reduced compared to those of the leakage of the Private Key. First the authentication credential restricts the attacker to the operations enabled by the Key Server whereas with the Private Key, the attacker is likely to perform any cryptographic operations without any restrictions. Second, by nature, the authentication credentials have internal cloud scope and can be specific for each container or each virtual machine, whereas the Private Key is used for the authentication outside the cloud and is shared by all containers or virtual machines. As a result, a revocation of the Private Key would impact all containers or virtual machines as well as TLS Client outside the cloud. On the other hand, the impact of revoking an authentication credential could be limited to a single container or virtual machine and without any impact for any TLS Clients outside the cloud. In a nutshell, this makes management of the authentication credentials much easier than management of the Private Key, and so much easier to mitigate a leakage.

In this scenario, the Key Server as well as the containers or virtual machine are expected to be hosted on the same Cloud. As a result, the latency between the Key Server and the containers or the virtual machine and the Key Server remains limited and acceptable for the TLS Client. In addition, the containers or virtual machines communicating with the Key Server may be different applications running on different operating systems.

## [5.2.](#) Content Provider Use Case

### [5.2.1.](#) Description

A Content Provider may use multiple Edge Servers that are directly exposed on the Internet. Edge Servers present a high risk of exposure as they are subject to, for example, misconfigurations as well as OS and web applications vulnerabilities at the Edge Server.

For example, as illustrated by the Heartbleed attack [[HEART](#)]. Specifically, the Heartbleed attack uses a weakness of a software implementation that allowed retrieval of the private key used by the TLS server. Such attack would not have been successful if the private key was not stored on the Edge Server. In addition, a Cloud Provider may run different implementations of web servers, or OS in order to make its infrastructure or service less vulnerable to a single point of failure. On the other hand, the diversity of implementations increases the risk of a leakage of the Private Key.

#### [5.2.2.](#) LURK Expectations

In this scenario, LURK enables a Content Provider to store the critical information in a more secured place such as, the Key Server, accessed only by all the authenticated Edge Servers.

Note that if the Private Key is shared between multiple Edge Servers, a leakage occurring at one Edge Server compromises all servers and the services. [Section 5.1](#) describes more in depth the difference between a leakage of the authentication credentials of the Edge Server versus the Private Key.

In this scenario, the Key Server is accessed by a limited number of Edge Servers which are authenticated. An Edge Server may present a vulnerability, it will not have access to the Private Key. It eventually may use the identity of the Edge Server to perform cryptographic operations with that Private Key, and means should be provided to limit the usability of such use.

In this scenario, the latency between the Edge Server and the Key Server depends on the distribution of the Edge Servers. When Edge Servers are far away from the Key Server, the time to set TLS Connection may be impacted by this architecture. In the event, such overhead impacts the quality of service of the TLS Client, the Content Provider may use multiple Key Servers to reduce the latency of the communication between the Edge Server and the Key Server. This scenario assumes that we are within a single administrative domain, so the Private Key remains inside the boundaries of such domain. In addition, the use of the Key Server prevents a direct access to the Private Key.

#### [5.3.](#) Content Owner / Content Provider Use Case



#### 5.3.1. Description

It is common that applications – like a web browser for example – use TLS to authenticate a Content Owner designated by a web URL and build a secure channel with that Content Owner.

When the Content Owner is not able to support all the TLS Client requests or would like to optimize the delivery of its content, it may decide to take advantage of a third party delivery service designated a Content Delivery Network (CDN) also designated as the Content Provider. This third party is able to locate the Edge Servers closer to the TLS Clients in various different geographical locations.

The Content Owner may still want to be authenticated by TLS Client while not terminating the TLS Connection of the TLS Client. In addition, while the Content Owner provides the Content Provider the content to deliver it may not agree or want to provide its Private Key to the Content Provider. In fact, the Private Key used to authenticate the Content Provider may present more value than the content itself. For example, the content may be accessed by devices or clients configured with the public authentication credential. In such cases, the leak of the Private Key and the renewal of the Private Key would require to configure all these devices. Such additional configuration are likely to affect the image of the Content Provider as well as result in some interruption of the service. The content, on the other hand may have only an ephemeral value and the Content Owner, may accept the risks of leakage and provide the Content Provider the content in cleartext. Alternatively, the content may also be encrypted with DRM, so its access remains restricted to authorized users only.

#### 5.3.2. LURK Expectations

In this scenario, LURK enables a Content Owner to delegate the delivery aspects while still controlling the authentication of the Content Owner. Similarly it enables the Content Provider to remain focused only on the delivery aspects of the content, without supporting the risks associated to the leakage of the Private Key. The Content Provider and the Content Owner are different administrative entities. As a result, the Key Server and the Edge Servers may be located in different networks and the communication between the Edge Server and the Key Server may introduce some delays. Such delay may be acceptable for the TLS Client, especially for long term TLS connections. Otherwise, the Content Owner, may provide the Key Server to the Content Provider. This use case is then very

---

similar to the one described in [Section 5.2](#). Note that providing the Key Server to the Content Provider in a hardware security module for example, still prevents the Content Provider to access the Private Key while it is able to serve content.

In this scenario, the Content Owner is likely to involve multiple Content Providers. In addition, the agreement between the Content Provider and the Content Owner may take various forms. The Content provider for example, may provide an infrastructure, or a delivery service. As a result, the Content Owner may not control the application or TLS library interacting with the Key Server.

#### [5.4](#). Content Distribution Network Interconnection Use Case

##### [5.4.1](#). Description

In the case of Content Distribution Network Interconnection (CDNI) [[RFC6707](#)], it may also that the company with which the Content Owner has contracted may further delegate delivery to another CDN with which the Content Owner has no official business relationship. Even if the Content Provider trusts the upstream CDN, and perhaps has strong legal contracts in place, it has no control over, and possibly no legal recourse against, the further downstream CDNs.

##### [5.4.2](#). LURK Expectations

In this scenario, LURK enables a delegation between CDNs of the delivery without providing the Private Key which would require additional agreement. As a result, LURK provides more agility in the delivery of content. Similarly to [Section 5.3](#) the delegating CDN may provide the content but not the Private Key. If the delegating CDN hosts the Key Server it needs to provide an access to the Key Server. On the other hand, the delegating CDN may not even host the Key Server, in which case, it may proxy the communications to the upstream CDN or the Content Owner. Other architecture may also enable a direct access to the Key Server by the delegated CDN.

In this scenario, different CDN are interacting, and the access to the Key Server may result in substantial additional latencies. This additional latency should not affect the quality of service of the delivery service. In addition, the motivations for providing content to the delegated CDN without providing the Private Key are similar to those of [Section 5.3](#).

## [6.](#) Requirements

The requirements listed in this section are limited to the LURK protocol, that is, the exchanges between the Edge Server and the Key Server.

### [6.1.](#) LURK Requirements

This section provides the requirements associated to the protocol used by the Edge Server and the Key Server. In the remaining section, this protocol is designated as LURK.

Multiple implementations of Edge Server, located in different administrative domains must be able to interact with multiple implementations of Key Servers also located in multiple administrative domains. In addition, the scope of LURK is closely related to TLS standardized at the IETF.

R1: LURK MUST be standardized at the IETF

LURK is limited to the Edge Server and the Key Server, so it is expected to be transparent to the TLS Client. In addition, in order to be deployed in the short term, any modification on the TLS Client should be avoided.

R2: LURK MUST NOT impact the TLS Client.

### [6.2.](#) Key Server Requirements

The Key Server holds the Private Key, and interacts with the Edge Servers.

R3: The Key Server MUST be able to provide the necessary authentication credential so the TLS Client and the Edge Server set an authenticate TLS Connection with the Private Key.

R4: The Key Server MUST NOT leak any information associated to the Private Key. In particular the Key Server MUST NOT provide a generic signing/encryption oracle.

R5: The Key Server SHOULD NOT perform any operation outside the authentication of a TLS Connection.

R6: The Key Server MUST provide confidential information to the Edge Sever over an authenticated and encrypted channel.

### [6.3.](#) Edge Server Requirements

R7: The Edge Server SHOULD be provisioned with the public authentication credentials. Note: Public certificate provisioning is outside of LURK.

## [7.](#) Security Considerations

LURK provides a centralized control of the Private Key. This centralized control was highly motivated by security to limit the risk of leakage of the Private Key while providing some flexibility and ability for different parties to interact together. On the other hand, centralizing the authentication at the Key Server provides the ability to control the access of content. More specifically, an entity that controls the Key Server is able to prevent an authenticated access to the content, by either not responding, or providing corrupted authentication credentials such as for example corrupted premasters to the Edge Server, or signatures that could not be checked by the TLS Client. Such global control of the access to content was not that easy when the content was distributed by a distributed network of Edge Servers. As a result such facility may be used by some governments agencies or make the Key Server an interesting target for attackers to control the access of the content. Note that even though the Key Server may remain inside an domain, as it is being accessed by Edge Servers, attackers may perpetrate attacks on the Key Server through these Edge Servers.

## [8.](#) IANA Considerations

There are no IANA considerations in this document.

## [9.](#) Acknowledgements

Thanks are due for insightful feedback on this document to Robert Skog, Hans Spaak, Salvatore Loreto, John Mattsson, Alexei Tumarkin, Yaron Sheffer, Eric Burger, Stephen Farrell, Richard Brunner, Stephane Dault, Dan Kahn Gillmor, Joe Hildebrand, Thomas Fossati, Kelsey Cairns.

## 10. References

### 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

Migault, et al.

Expires December 30, 2016

[Page 11]

---

Internet-Draft

LURK/TLS Use Cases

June 2016

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.

### 10.2. Informative References

- [HEART] Codenomicon, "The Heartbleed Bug", <<http://heartbleed.com/>>.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", [RFC 6707](#), DOI 10.17487/RFC6707, September 2012, <<http://www.rfc-editor.org/info/rfc6707>>.

## Authors' Addresses

Daniel Migault  
Ericsson  
8400 boulevard Decarie  
Montreal, QC H4P 2N2  
Canada

Phone: +1 514-452-2160  
Email: [daniel.migault@ericsson.com](mailto:daniel.migault@ericsson.com)

Kevin Ma J  
Ericsson  
43 Nagog Park  
Acton, MA 01720  
USA

Phone: +1 978-844-5100  
Email: kevin.j.ma@ericsson.com

Rich Salz  
Akamai

Email: rsalz@akamai.com

Sanjay Mishra  
Verizon Communications

Email: sanjay.mishra@verizon.com

Migault, et al.

Expires December 30, 2016

[Page 12]

---

Internet-Draft

LURK/TLS Use Cases

June 2016

Oscar Gonzales de Dios  
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

