Kiuru MSSP 6.0

KeyURI Specification

Document version 0.9

Change History

Version	Date	Changes
0.1	2024-02-11	Initial version
0.6	2024-03-26	Defined the URI scheme more clearly. Added new properties.
0.7	2024-03-26	Added comparison chapter. Added more properties.
0.8	2024-03-26	Cleanup
0.9	2024-03-27	Added "local" LoA scheme

Methics Oy
Stella Business Park
Lars Sonckin Kaari 14
FI-02600 Espoo
Finland

Business ID: 1778566-8

Table of Contents

1 Introduction

This document acts is specification of the keyuri: URI scheme, which is also known as KeyURI. The KeyURI is used to identify end entity's private keys in a secure signature creation device. The KeyURI is used when an app access to the private keys.

Additionally, this document contains KeyURI version model and design for various components closely related to the KeyURI.

1.1 Audience

KeyURI integrators and developers.

1.2 Typographic Conventions

Example	Description
`conf/example.conf`	File names are shown in mono-spaced font.
`$ export`	Typed commands are shown in mono-spaced font and preceded by a dollar-sign or other prompt. If the prompt is "#", it refers to a super-user executable command.
`kiuru.service.connections = 800`	Configuration examples are shown in a mono-spaced font, just like typed commands. Due to typographic reasons, texts have sometimes folded on multiple lines and there might either be no indication of this, or a backslash "\" at the end of the row highlighting that the text continues on the next row.

`$ export …`
`$ cp …`
`$ restart …`
	Configuration and command examples which span several rows are highlighted with gray background color.

`$ export …`	Commands that begin with `$` can be run as a non-root user.
`# su - …`	Commands that begin with `#` must be run as a root user.
...see chapter *15 Configuration Options* for more info...	References to other sections, documents or other resources are shown in italics.
<ap_id>	User input values are italic and contained in angle brackets.

2 Overview

KeyURI is an identifier that serves the following purposes:

1. to uniquely identify a key
2. to allow an application provider to request a specific key from an application
3. to list key properties

2.1 URI Definition

The following RFCs define the syntax and usage

- URI: Generic Syntax https://www.rfc-editor.org/rfc/rfc3986
- URI Design and Ownership https://www.rfc-editor.org/rfc/rfc8820
- Well-known URIs https://www.rfc-editor.org/info/rfc8615

Additional information is in URI fragment as defined in
https://www.rfc-editor.org/rfc/rfc3986#section-3.5

Only the URI specification defines the fragment as required in
https://www.rfc-editor.org/rfc/rfc8820#name-uri-fragment-identifiers

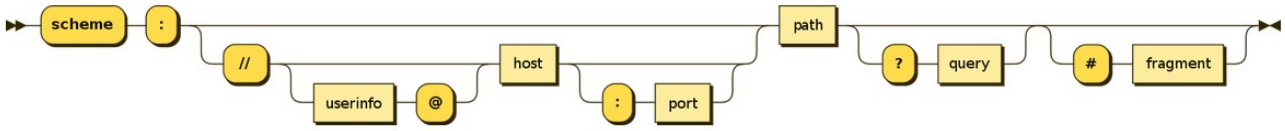

Figure 1: URI syntax diagram

3 KeyURI Definition

3.1 KeyURI: Scheme

KeyURI is an URI scheme similar to the PKCS#11 URI scheme (RFC 7512). The path parameter of KeyURI is currently always "key", and the scheme is "keyuri". All properties are defined as query parameters.

Example:

```
keyuri:key?usage=authn&loa=eidas-high&provider=Test&sscd=sim&key-algorithm=rsa
```

Each property may have multiple values if needed. Values are separated by comma. Example:

```
keyuri:key?usage=authn,signing&
provider=Test&sscd=sim&key-algorithm=rsa&usage=authn
```

The KeyURI scheme defines the following key properties:

Property	Description	Multivalue
usage	Key Usage. One of: "authn", "signing"	no
sscd	SSCD type (e.g. "sim")	no
provider	Key provider. This can be for example an MNO brand name ("DNA").	yes
loa	Level of Assurance (e.g. "eidas-high")	yes
identity-scheme	Identity scheme used by the signature service. E.g. NIST or eIDAS.	yes
key-id	Key identifier. This should be globally unique per-key.	no
key-name	Name of the key. This can be set e.g. by the key generator.	no
key-algorithm	Algorithm of the key. Typically "ecc" or "rsa".	no
key-length	Length of the key. E.g. 2048 for RSA 2K	no
key-pregenerated	Is the key pre-generated? true/false	no
rsa-public-exponent	RSA public exponent. Only relevant when key algorithm is RSA.	no
ec-curve	ECC curve of the key. Relevant only for elliptic keys. Name or OID.	no
created-date	ISO-8601 timestamp of the key creation time	no

3.2 Key Usages

Supported Key Usages include, but not limited to:

Key Usage	Description
authn	X.509 KeyUsage digitalSignature
signing	X.509 KeyUsage nonRepudiation
keyEncipherment	X.509 KeyUsage keyEncipherment
dataEncipherment	X.509 KeyUsage dataEncipherment
keyAgreement	X.509 KeyUsage keyAgreement
keyCertSign	X.509 KeyUsage keyCertSign
cRLSign	X.509 KeyUsage cRLSign
encipherOnly	X.509 KeyUsage encipherOnly
decipherOnly	X.509 KeyUsage decipherOnly

3.3 Level of Assurance

Level of Assurance (loa) is a property that defines the key's trustworthiness. A key can have multiple different assurance levels on different schemes. For example, a key could be considered "loa4" by ISO standards, but only "substantial" by EIDAS standards.

KeyURI allows specifying multiple LoA with a scheme prefix.

Example:

```
keyuri:key?loa=eidas-substantial,iso-high&
provider=Test&sscd=sim&key-algorithm=rsa2k&usage=authn
```

3.3.1 LoA Schemes

Known Level of Assurance schemes are:

Prefix	Scheme	Description
iso	ISO-29115	ISO/IEC 29115:2013 ("LoA1", "LoA2", "LoA3", "LoA4")
eidas	EIDAS-2014	Regulation *(EU) No 910/2014* ("low", "substantial", "high")
nist	NIST-SP-800	NIST Special Publication 800-63-3 ("IAL1", "IAL2", "IAL3)
local	Local	Any country-specific LoA scheme.

Additional schemes can be used if required.

3.4 SSCD Types

SSCD type gives a hint on what type of an SSCD the key is located in.

Supported SSCD types include:

Type	Description
sim	UICC PKI cards
remote	ETSI Remote Signature
local	Local keystore
nfc	Local NFC connected SSCD
other	Other unspecified keystore

4 KeyURI Comparison

KeyURI comparison is useful when trying to search matching keys. To compare two KeyURIs, the following rules should be followed:

1. Ignore all properties that are not present in both KeyURI
2. Convert both KeyURI to lower case to avoid case-sensitivity issues
3. Check that each property that is not multi-valued matches
4. Check that each value in a multi-valued property exists in the comparison KeyURI

Example:

```
KeyURI:
keyuri:key?key-algorithm=rsa,key-length=2048,loa=eidas-high,nist-ial4,sscd=sim

Comparison:
keyuri:key?key-algorithm=rsa,loa=eidas-high
```

The above comparison KeyURI is a match because:

1. Both contain key-algorithm and loa
2. KeyURI contains the key-algorithm required by the comparison KeyURI