

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 7, 2016

M. Miller  
S. Nandakumar  
Cisco Systems, Inc.  
July 06, 2015

Key Discovery Service  
draft-miller-saag-key-discovery-00

Abstract

A typical requirement with any cryptographic key management system is to provide discovery, retrieval, distribution, and management of keys across entities needing to perform the necessary security operations. However there exists no standard mechanism to automatically discovery the keys, but rather the keys are either provisioned statically or shared beforehand via non standard mechanisms. This document defines machanisms for an entity to automatically discover the key(s) associated with other entities using the WebFinger protocol.

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 January 7, 2016.

Copyright Notice

Copyright (c) 2015 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

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Locating an Entity's Public Keys . . . . .	3
4. Locating a Resource's Key . . . . .	5
5. IANA Considerations . . . . .	6
5.1. "key:" URI Scheme . . . . .	6
5.1.1. URI Scheme Name . . . . .	7
5.1.2. Status . . . . .	7
5.1.3. URI Scheme Syntax . . . . .	7
5.1.4. URI Scheme Semantics . . . . .	7
5.1.5. Encoding Considerations . . . . .	7
5.1.6. Applications/Protocols That Use This URI Scheme Name . . . . .	7
5.1.7. Interoperability Considerations . . . . .	8
6. Security Considerations . . . . .	8
7. References . . . . .	8
7.1. Normative References . . . . .	8
7.2. Informative References . . . . .	9
Appendix A. Determining a URI from Encrypted Content for Key Discovery . . . . .	9
Authors' Addresses . . . . .	10

## 1. Introduction

With the increase in efforts towards ensuring end to end encryption for communications on the Internet, it has become necessary to improve the experience around how cryptographic primitives such as keys and certificates are discovered, distributed, and managed. Efforts such as [I-D.barnes-acme] attempts to automate aspects of certificate retrieval and management, whereas efforts such as [I-D.abiggs-saag-key-management-service] provides mechanisms for dealing with keys required for secure group communications. However, today's standard efforts lack mechanisms for easy discovery of keys associated with an entity or a resource on the Internet. For example, any public key cryptography based system relies on being able to have acquired the public key(s) of the target entity in order to establish a secure communication with that entity. For these scenarios, the entities wanting to acquire such keys are either provisioned with the keys statically (as part of the configuration) or distributed by non standard (application specific) means.

This document describes mechanisms for entities to automatically discover the cryptographic keys associated with entities (users/

resources) using WebFinger [RFC7033] as the protocol mechanism. Such a mechanism provides an added benefit of separating key discovery from its retrieval and management.

The rest of this document is organized as follows. Section 3 shows using WebFinger protocol for entity's public key using an 'acct' URI [RFC7565], followed by Section 4 showing the same procedure for retrieving a secret key for a file resource.

## 2. Terminology

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].

## 3. Locating an Entity's Public Keys

The examples below show query and response on the WebFinger resource for retrieving the public key(s), using an 'acct' URI [RFC7565]:

Query WebFinger:

```
GET /.well-known/webfinger?  
    resource=acct%3Aabilbo.baggins%40hobbiton.example  
    HTTP/1.1  
Host: hobbiton.example
```

The WebFinger response then includes links to the entity's public keys:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: hobbiton.example
Access-Control-Allow-Methods: GET OPTIONS
Content-Type: application/jrd+json

{
  "subject": "acct:bilbo.baggins@hobbiton.example",
  ...
  "links": [
    ...
    {
      "rel": "public-key",
      "href": "https://hobbiton.example/~bilbo.baggins/
              pubkeyset.json",
      "type": "application/jwk-set+json"
    }
  ]
}
```

The "rel" value is 'public-key'. The "href" MUST be a HTTPS URI that the entity's public key(s) is retrieved from, formatted as a JWK or JWK-set (as defined in [RFC7517]):

```

{
  "keys": [
    {
      "kty": "EC",
      "kid": "bilbo.baggins@hobbiton.example",
      "use": "sig",
      "crv": "P-521",
      "x": "AHKZLL0sCOzz5cY97ewNUajB957y-C-U88c3v13nmGZx6sYl_o
      JXu9A5RkTKqjqvjyekWF-7ytDyRXYgCF5cj0Kt",
      "y": "Adym1HvOiLxXkEhayXQnNCvDX4h9htZaCJN34kfmC6pV5OhQHi
      raVySsUdaQkAgDPrwQrJmbnX9cwlGfP-HqHZR1"
    },
    {
      "kty": "RSA",
      "kid": "bilbo.baggins@hobbiton.example",
      "use": "sig",
      "n": "n4EPtAOCc9AlkeQHPzHStgAbgs7bTZLwUBZdR8_KuKPEHLd4rH
      VTeT-O-XV2jRojdNhxJWTDvNd7nqQ0VEiZQHz_AJmSCpMaJM RB
      SFKrKb2wqVwGU_NsYOYL-QtiWN2lbzcEe6XC0dApr5ydQLrHqk
      HHig3RBordaZ6Aj-oBHqFEHYpPe7Tpe-OfVfHd1E6cS6M1FZcD
      lNNLYD5lFHpPI9bTwJlSde3uhGqC0ZCuEHg8lhzwOHrtIQbs0F
      Vbb9k3-tVTU4fg_3L_vniUFAKwuCLqKnS2BYwdq_mzSnbLY7h_
      qixor7jig3__kRhuaxwUkRz5iaiQkqgc5gHdrNP5zw",
      "e": "AQAB"
    },
    {
      "kty": "RSA",
      "kid": "bilbo.baggins@hobbiton.example",
      "use": "enc",
      "e": "AQAB",
      "n": "uTWZBa8bjLQNJ9cBrdxGV_H_pmHEDuAXpCR1NnyYQYkUGJ8F3a
      y_OM6sw82fS2ZcAXHpCVYlp30pd4D6BYwwixDt_eSkY-NLhPA3
      ouE4YwtaUVZYBZT909pISRK4WOr3nXeJ01ltrgPQ7StBR1C776
      KJnsHbBPdXO7tpAfph9GnjNUJxrpoFmhiZx3hbpeUpxsTsDuB9
      doVN9cFCpsjPpoiAvkr_DoyckbilTnR4zwzDQyfSkhNYghFuqh
      vAQQ8yMQ29HOHYdfON2Z8yCjgAnyJCS1lnywkYaAaZGyxhozXr
      F6_Np2BHteL_XRNekhY72gtlnRZYCQArjJMACx_3iw"
    }
  ]
}

```

#### 4. Locating a Resource's Key

The example below shows WebFinger query and response for retrieving the secret key associated with a resource controlled by erebor.com, identified using a 'key' URI scheme defined in Section 5.1. The URI to query could have been determined as per Appendix A.

Query WebFinger:

```
GET /.well-known/webfinger/  
    ?resource=key%3Asha-256.GJa85ytSaKlpX6uwyBIEZFRLn5ZjrDd36emx  
    NmAGP_s@erebor.eample  
    HTTP/1.1  
Host: erebor.example
```

WebFinger Response: The "rel" value is 'secret-key'. The "href" indicates where to retrieve the secret key.

```
HTTP/1.1 200 OK  
Access-Control-Allow-Origin: hobbiton.example  
Access-Control-Allow-Methods: GET OPTIONS  
Content-Type: application/jrd+json
```

```
{  
  "subject": "key:sha-256.GJa85ytSaKlpX6uwyBIEZFRLn5ZjrDd36emxNmAGP_s  
             @erebor.eample",  
  ...  
  "links": [  
    ...  
    {  
      "rel": "secret-key",  
      "href": "kms://rivendell.example/key/  
              c8e84a7d-2ae1-435a-9738-bb00e4c8dc7a",  
      "type": "application/jwk+json"  
    }  
  ]  
}
```

If "href" is an HTTPS URI, the type SHOULD be "application/jwk+json" or "application/jwk-set+json". Other protocols might use different container formats.

## 5. IANA Considerations

### 5.1. "key:" URI Scheme

In accordance with the guidelines and registration procedures for new URI schemes [RFC4395], this section provides the information needed to register the 'key' URI scheme.

#### 5.1.1. URI Scheme Name

key

#### 5.1.2. Status

permanent

#### 5.1.3. URI Scheme Syntax

The 'key' URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [RFC5234], borrowing the 'host' and 'unreserved' rules from [RFC3986]:

```
keyuri = "key" ":" keyid "@" host
keyid  = 1 * unreserved
```

#### 5.1.4. URI Scheme Semantics

The 'key' URI scheme identifies cryptographic keys provided by organizations, identified by domain name. It is used only for identification, not for interaction. A protocol (other than the one specified in this document) that employs the 'key' URI scheme is responsible for specifying how a 'key' URI is dereferenced in the context of that protocol.

#### 5.1.5. Encoding Considerations

- o The keyid consists of unreserved characters as defined in [RFC3986].
- o The host consists only of Unicode code points that conform to the rules in [RFC5892].
- o Internationalized domain name (IDN) labels are encoded as A-labels [RFC5890].

#### 5.1.6. Applications/Protocols That Use This URI Scheme Name

At the time of this writing, only this protocol uses the 'key' URI scheme, in conjunction with WebFinger. However, use is not restricted to this protocol, and the scheme might be considered for use in other protocols.

#### 5.1.7. Interoperability Considerations

There are no known interoperability concerns related to the use of the 'key' URI scheme.

#### 6. Security Considerations

As this document is in essence a profile of WebFinger [RFC7033], all of the security considerations from that draft apply.

Because anyone with the symmetric secret key can use it for decryption, access to symmetric secret keys SHOULD require authorization. Such authorization enforcement SHOULD be at the URI for the key, and MAY also be enforced on the WebFinger query.

#### 7. References

##### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, February 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.
- [RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "WebFinger", RFC 7033, September 2013.



## 7.2. Informative References

- [I-D.abiggs-saag-key-management-service]  
Biggs, A. and S. Cooley, "Key Management Service Architecture", draft-abiggs-saag-key-management-service-02 (work in progress), July 2015.
- [I-D.barnes-acme]  
Barnes, R., Eckersley, P., Schoen, S., Halderman, A., and J. Kasten, "Automatic Certificate Management Environment (ACME)", draft-barnes-acme-02 (work in progress), May 2015.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, May 2015.
- [RFC7565] Saint-Andre, P., "The 'acct' URI Scheme", RFC 7565, May 2015.

## Appendix A. Determining a URI from Encrypted Content for Key Discovery

In most cases, the URI on which to perform key discovery will be known. Chat rooms, conferencing services, and even shared files oftentimes have a URI for addressing the resource. Occasionally protected content will be disseminated in a manner that an explicit URI cannot be known or conveyed, but a domain name for where the content originated from might be known. The following is an algorithm that can be used to determine a URI for discovering the key in such cases.

1. Start with the encrypted content, C.
2. Perform a SHA-2 [RFC6234] hash (e.g., SHA-256) over the encrypted content, to produce I'.
3. Perform the URL-safe Base64 encoding [RFC4648] over I' to produce I.
4. Concatenate the following to produce the URI for key discovery, U:
  - \* The scheme "key:";

- \* The name of the hash used in Step 2 (as registered in the IANA Hash Function Textual Names registry; e.g., "sha-256"), H;
- \* The character "." (U+002E FULL STOP);
- \* The base64url-encoded hash from Step 2, I;
- \* The character "@" (U+0040 COMMERCIAL AT); and
- \* The domain name, D

Expressed as an algorithm:

```
U := "key:" || H || "." || BASE64URL(SHA2(C)) || "@" || D
```

For example, suppose one has some encrypted content for which they do not have the key, but is known to come from "erebor.example". If the SHA-256 hash of the encrypted content were (in hex):

```
1896bce72b5268ad695fab0c8120464544b9f9663ac3777e9e9b13660063feb
```

The URI to use for key discovery is then:

```
key:sha-256.GJa85ytSaKlpX6uwyBIEZFRLn5ZjrDd36emxNmAGP_s@erebor.eample
```

From here, the receiver of the encrypted content uses the calculated URI to perform key discovery for a resource as described in Section 4.

#### Authors' Addresses

Matthew Miller  
Cisco Systems, Inc.

Email: mamille2@cisco.com

Suhas Nandakumar  
Cisco Systems, Inc.

Email: snandaku@cisco.com