

Network Working Group  
Internet-Draft  
Intended status: Best Current Practice  
Expires: August 18, 2014

H. Tschofenig  
ARM Ltd.  
February 14, 2014

Authentication and Authorization for Constrained Environments (ACE):  
Overview of Existing Security Protocols  
draft-tschofenig-ace-overview-00.txt

Abstract

This document surveys existing three party authentication and authorization protocols for use with Internet of Things use cases. The discussed protocol frameworks are Kerberos, OAuth, ABFAB, and the certificate model. The aim is to understand whether any of the available standardized security protocols are re-usable for constrained environments. A future version of this document will provide a more detailed analysis against the requirements.

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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 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.

## 1. Introduction

[I-D.seitz-ace-usecases] introduces a number of use cases that require device-to-device authentication whereby both devices may be constrained. [I-D.ietf-lwig-terminology] discusses the different types of constraints of these devices.

This document aims to raise the high-level question about the possible re-use of existing three party authentication and key exchange protocols for use in IoT environments. This version of the document does not aim to map requirements derived from the use cases against these protocols. Such a detailed analysis is premature at this point when use case descriptions are still in flux.

The starting assumption for the architectures in this document is that a device (a client) wants to access some resource (referred as service in this document). It unfortunately does not have any relationship with the server offering that service. Figure 1 shows the scenario graphically.

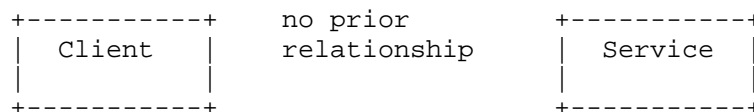


Figure 1: Two Party Scenario.

Imagine that the client is a light-switch and the service is a light-bulb.

Today, companies solve this case by using a pairing protocol (at the link layer typically) where the two devices execute a special imprinting/pairing protocol to establish an initial key by using out-of-band (OOB) channel. This OOB channel can come in many forms:

- o Using an alternative communication channel, such as a USB stick, Ethernet cable
- o Human involvement by comparing hashed keys, entering passkeys, scanning QR codes
- o Second wireless connectivity (e.g., infrared)

- o Proximity-based information

The pairing is a suitable approach where wireless communication replaces a wired communication technology previously used. For example, a headset connected to a music player using a wired connection is replaced with the wireless version. Not all use cases do, however, allow users to pair their device with other devices upfront. Consider an enterprise with electronic door locks. It is hard to imagine an employee who has to pair their digital key with every door in the building first before they use the system.

Requiring every device to pair with every other device upfront is often inconvenient or not feasible. Hence, this document does not explore pairing solutions further. To offer an improved user experience with better scalability properties a device might either share credentials with some trusted third party. There are various ways how credentials can be shared with these trusted third parties. For example, credentials may be provisioned during the manufacturing process or devices may have been paired with the trusted third party (in case the trusted third party is local to the user). In fact, today it is very common for IoT devices to have at least credentials pre-provisioned for use with the vendor / manufacturer of the device to allow software updates to be provided securely.

Thus, we move to a model where the device (client) shares some credentials with a trusted third party. This trusted third party does not need to be a server on the Internet; ideally it could also be operated locally within someones' home, within an enterprise, or within a factory.

This three party architecture is shown in Figure 2.

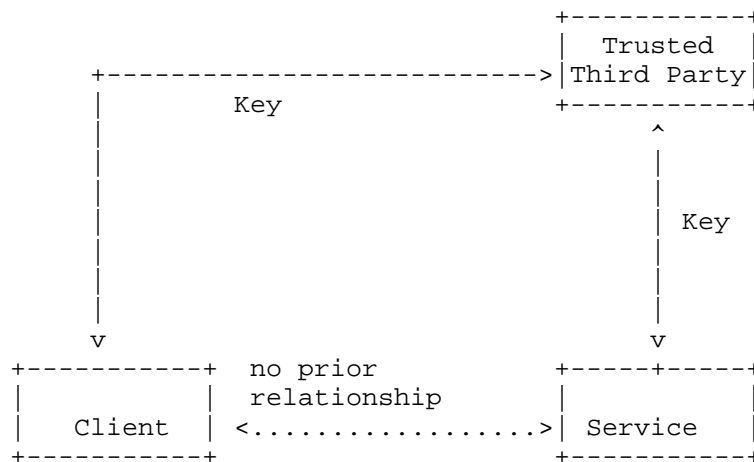


Figure 2: Three Party Scenario.

This three party architecture and messaging pattern has been explored with prior IETF work and this document lists the most relevant efforts (on a high level).

The goal of the communication exchange is that the client has been authorized to access the service, and is able to secure the exchange of information. The client and the service may, optionally, possess keying material for future use of the service with the benefit of better performance for future interactions.

Note: This document does not aim to cover the use cases in their entirety. First, we assume that the security protocol interaction for link layer authentication is outside the scope. The focus of this document is on the application layer interactions when accessing services. Second, this document does not survey access control policy languages and mechanisms for managing these access control policies. These policies are important since many of the systems described below only provide an answer to the question 'Who is the holder of this key?' and standards for answering the question 'Can this key be used for this purpose?' (authorization) are often realized in a proprietary way.

While Figure 1 shows three parties the protocols described in Section 2 have been generalized to four or even multi-party scenario. The result is shown in Figure 2.

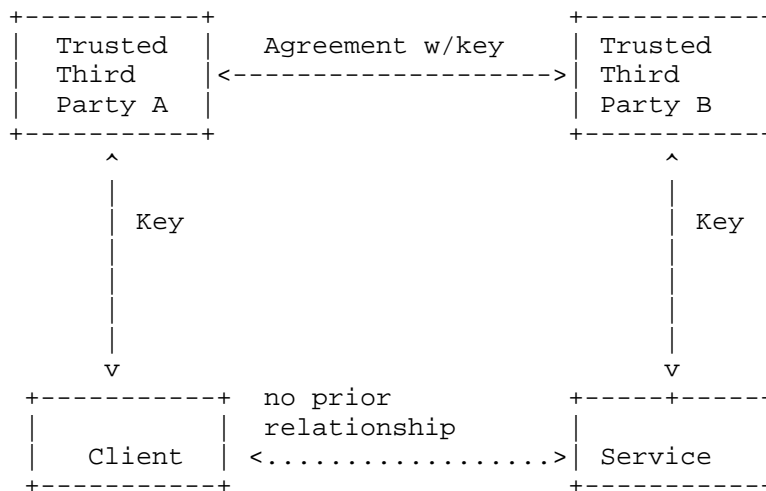


Figure 3: Generalization of Three Party Scenario.

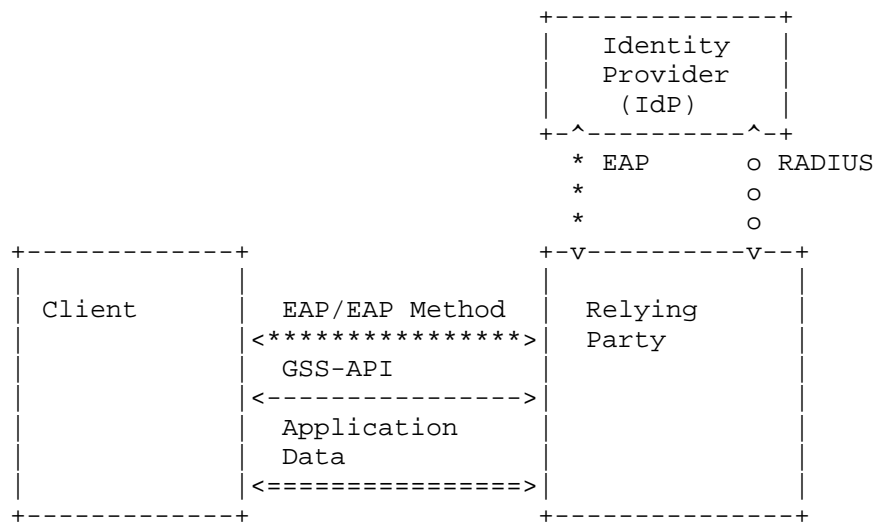
## 2. Three Party Security Frameworks

This section introduces four authentication and authorization frameworks standardized in the IETF. The description is intentionally kept at a high level and a reader is encouraged to consult the referenced documents for details and various options these protocols offer. The terminology with each of these protocols is lightly different and appropriate mappings have been applied.

To demonstrate the level of maturity of these frameworks availability of products, source code, and deployment experience is mentioned for each of these frameworks. Note, however, that this experience does not imply suitability for use with the IoT environment.

### 2.1. Application Bridging for Federated Access Beyond Web Architecture

This section describes the Application Bridging for Federated Access Beyond Web architecture [I-D.ietf-abfab-arch], which builds on the Authentication, Authorization and Accounting (AAA) framework. The AAA framework re-uses the Extensible Authentication Protocol (EAP) [RFC5247] and EAP methods for the authentication protocol capabilities. A detailed description of the AAA keying framework can be found in [RFC5247].



#### Terminology Mapping:

- The term 'Relying Party' corresponds to the 'service'.
- The term 'Identity Provider' corresponds to the 'trusted third party'.

Figure 4: ABFAB Architecture.

With the message exchange shown in Figure 4 the client wants to obtain access to a service and starts interacting with that service. Since no prior relationship between the client and the service is assumed the EAP message exchanges is relayed by the service and the EAP server component of the IdP. Between the client and the service these EAP payloads are encapsulated within the GSS-API. After a successful authentication and authorization session keys are delivered from the IdP to the service and can then be used to secure the application layer data exchange between the client and the service.

While the use of EAP and the AAA architecture has mostly found use for network access authentication the work on ABFAB applies this architecture to application layer services.

#### Pros:

- o Re-uses existing protocols: RADIUS, GSS-API, EAP, EAP methods.

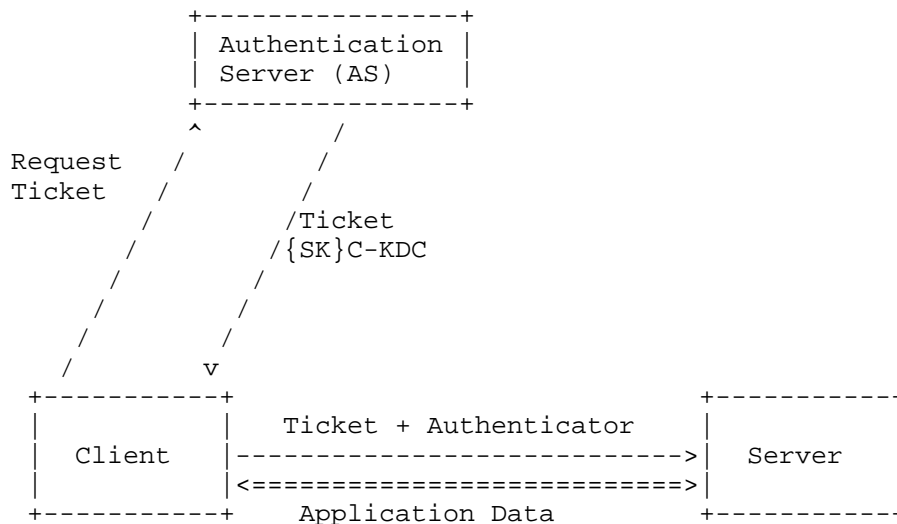
- o Security properties of the AAA / EAP framework well studied and large deployments of the AAA framework exist.
- o Products and open source code exists for EAP, EAP methods, RADIUS, and the GSS-API. The extensions needed for ABFAB also have been implemented but they are less mature compared to the EAP/AAA deployment.
- o Large range of EAP methods available offering all possible authentication and key exchange protocols for authentication between the client and the AAA server. These mechanisms have been deployed and are in widespread use today. While many EAP methods have been standardized only a few are in widespread use in non-IoT environments. However, there are many (open source) implementations available such that further experience concerning IoT suitability can be gathered.
- o IoT devices might use the AAA/EAP architecture for network access authentication (e.g., WLAN-based, IEEE 802.15.4-based ZigBee-IP deployments).
- o The AAA framework also supports authentication in a federated environment.
- o Authorization information is conveyed within RADIUS (and potentially in SAML assertions, as envisioned by ABFAB).

Cons:

- o The initial authentication and authorization exchange requires real-time interaction between the AAA server and the service.
- o Deployments have so far used this architecture mainly for network access and for specific applications (VoIP) only. Experience with other applications, as ABFAB envisions, is rather limited.
- o ABFAB architecture uses layering of EAP within the GSS-API, which adds additional overhead. A binding for the transport of EAP payloads in CoAP, for example, does not exist.
- o No unified authorization policy language has been defined for the AAA/EAP architecture. Instead, RADIUS attributes carry information about access control decisions.

## 2.2. Kerberos

Kerberos [RFC4120] is authentication system for distributed environments that has enjoyed deployment for more than three decades. The security properties have been extensively studied and various implementations exist.



### Terminology Mapping:

- The term AS corresponds to the 'trusted third party.'
- The term Server corresponds to the 'service'.

Figure 5: Kerberos.

The Kerberos exchange shown in Figure 5 illustrates a client who wants to get access to a server. It first has to interact with the Authentication Server (AS) to request a ticket. In response, the AS provides a ticket, which is a data structure encrypted with a key known only between the server and the AS. This ticket includes information about the client, a session key (SK) for later use, and various other security relevant information elements. The client also obtains the session key encrypted with a key it shares with the AS.

When a service access is required then the client interacts with the server and presents the ticket along with an Authenticator. The Authenticator demonstrates that the client was able to decrypt the session key with the key it shares with the AS and that it was able



to apply this key to compute a keyed message digest over several fields, including a time-stamp, when accessing the service. The time-stamp avoids replay attacks.

Pros:

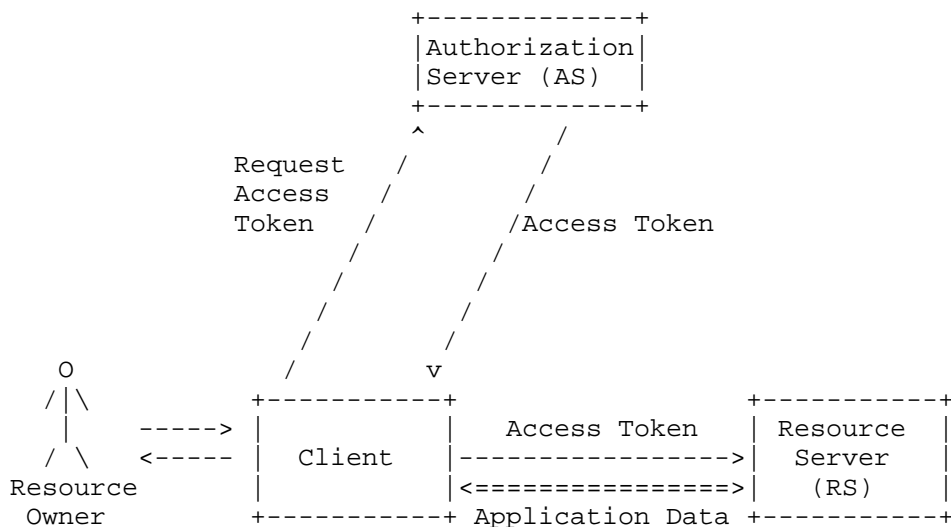
- o Re-uses existing protocol: Kerberos
- o Security properties well studied and large deployments exist.
- o Products and open source service exist.
- o Most parts of Kerberos, particularly the ticket concept, are designed with symmetric key cryptography, which improves performance. The Kerberos ticket is consequently fairly small and uses a binary encoding.
- o Kerberos also supports cross-realm authentication for scalable deployments.
- o Kerberos also specifies a UDP-based transport.
- o The message exchanges between the client and the service can be tailored to the need of the application.

Cons:

- o Each ticket is only usable for a single service (intentionally). As such, new tickets have to be requested whenever the client wants to access a new service or when the ticket expired.
- o For the authentication between the client and the KDC a limited number of authentication protocols have been specified.
- o Kerberos uses ASN.1 for encoding of the ticket and various messages. This may increase implementation complexity but the binary encoding is more efficient than other encodings, like XML or JSON.
- o No standardized access control policy has been standardized for inclusion inside a ticket. Proprietary policies are, however, used in real-world deployments.
- o A CoAP binding for the KRB\_PRIV and the KRB\_SAFE message exchanges used to secure application data between the client and the service have not been defined.

### 2.3. OAuth

The OAuth protocol is a recent development for the Web, which re-uses the Kerberos interaction pattern with influences from the Web / mobile app space. It initially aimed to solve the problem of delegated access to protected resources where websites asked users to share their long-term password. Over time OAuth has been used in other use cases that require delegated access.



#### Terminology Mapping:

- The term AS corresponds to the 'trusted third party'.
- The term RS corresponds to the 'service'.

Figure 6: Simplified OAuth Architecture.

Figure 6 shows the high-level OAuth message exchange. The canonical OAuth example allows a web user (resource owner) to grant a printing service (client) access to her private photos stored at a photo sharing service (resource server), without sharing her username and password. Instead, she authenticates directly with the authorization server which issues the printing service delegation-specific credentials.

#### Pros:

- o Re-uses existing protocols: OAuth Core [RFC6749], OAuth Bearer Token [RFC6750] OAuth MAC Token [I-D.ietf-oauth-v2-http-mac]/ HOTK [I-D.tschofenig-oauth-hotk], JWT [I-D.ietf-oauth-json-web-token].
- o Large deployments in the Web environment exist, which use the OAuth Bearer Token.
- o Products and open source service exist.
- o OAuth is flexible with regard to the used cryptography. A standardized format for the access token has been described with the JSON Web Token (JWT). For security protection of the JWT various specifications from the JOSE working group are available.
- o The message exchanges between the client and the service can be tailored to the need of the application. Bindings are available for HTTPS and SASL [I-D.ietf-kitten-sasl-oauth].
- o With regard to the offered security mechanism the interaction between the client and the resource server gives several choices: The OAuth Bearer Token requires a TLS exchange between the client and the resource server. The MAC Token specification is conceptually similar to Kerberos; a version based on asymmetric cryptography exists as well (see HOTK).

Cons:

- o For an environment with more than one authorization server or where the authorization server is located in a different domain than the resource server the standardization work is still in progress. Efforts have mostly be done in Kantara with the User-Managed-Access (UMA) working group.
- o A binding for CoAP does not exist for the client to authorization server nor for the client to resource server.
- o The OAuth architecture does not standardize the authentication procedure of the resource owner to the authorization server itself. This is a common approach for the Web environment where a number of different authentication protocols are in use in the browser. As such, the protocol works with any authentication mechanism.

## 2.4. Certificate Model

Prior work on the Public Key Infrastructure, certificate formats, certificate extensions, and various certificate management protocols can be re-used in the IoT context. With respect to the use cases

described in [I-D.seitz-ace-usecases] certificates may be short-lived and might need to contain attributes (which may be used for making access control decisions) rather than purely relying on the identity of users and their devices.

For the purpose of dynamic provisioning short-lived certificates, this document envisions to re-use a subset of the functionality offered by protocols like the Certificate Management over CMS (CMC) [RFC5272], the Certificate Management Protocol (CMP) [RFC4210], the Simple Certificate Enrollment Protocol [I-D.nourse-scep], Certification Request Syntax Standard - PKCS#10 [RFC2315] (with TLS or with PKCS#7 [RFC2986] ). While these protocols offer slightly different features, on a high-level they all fulfill the same function. Note that the management of trust anchors may be provided by a different protocol, such as Trust Anchor Management Protocol (TAMP) [RFC5934].

Of course, certificates do not necessarily need to be short-lived and could even be provisioned during the manufacturing process and never changed during the lifetime of the device. The drawback of such an approach is, however, that mechanisms for certificate revocation have to be provided. Furthermore, privacy concerns might arise since the same client certificate content will be shown to every service rather than information that is only relevant for a specific purpose.

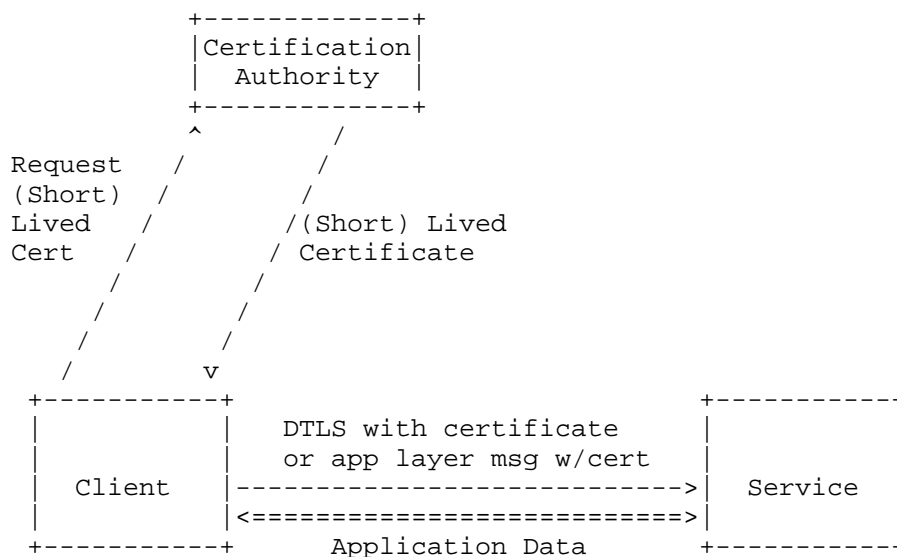


Figure 7: Certificate Model.

## Pros:

- o Re-uses existing protocols: DTLS (or application protocol), CMP/CMC/PKCS#10/SCEP, specifications (certificate format - RFC 6818 [RFC6818]), and concepts (PKI).
- o Large deployments on the Web and with enterprise system exist.
- o Products and open source code exists.
- o The certificate format offers flexibility in terms of content. New extensions have been defined over time.
- o Certificates can be used with DTLS without any additional modifications. Certificates can also used with application security mechanisms.
- o Authorization information may be placed in an extension of a public key certificate or in a separate attribute certificate [RFC3281]. Earlier work on KeyNote [RFC2704] could be re-used as it provides a more flexible authorization policy language.
- o A single certificate can be used with a number of different services.
- o Various PKI management protocols have been defined and they offer some flexibility. The properties vary on the specific use cases.

## Cons:

- o The certificate format and the PKI management protocols use ASN.1.
- o No UDP or CoAP transport is defined for CMC/CMP/SCEP. For PKCS#10 no transport is defined at all.
- o The public key infrastructure only focuses on asymmetric cryptography. A separate body of work is available for provisioning symmetric keys (like one-time-keys), such as the Portable Symmetric Key Container (PSKC) [RFC6030] and Dynamic Symmetric Key Provisioning Protocol (DSKPP) ([RFC6063]).
- o Protocols for certificate enrollment are in use but many deployments use their own strategy for distributing certificates (typically long-lived) to their users.
- o Asymmetric cryptography is computationally more expensive than symmetric cryptography but offers additional security benefits.

### 3. Conclusion

Several existing protocols can be used to meet the use cases outlined in [I-D.seitz-ace-usecases]. Each technology presented here offers a number of possibilities for profiling to make them work on for constrained devices. Despite the range of available security protocols, the use cases suggest that there is a need to profile and to extend those in order to make them fit for the constrained environment.

The right choice of authentication and authorization protocol will heavily depend on the envisioned usage environment.

It is, however, also worth noting that several aspects that are not discussed in this document although they appear as requirements in the use case document, namely

- o a language for describing access control policies,
- o the encoding of these policies, and
- o the container for associating these policies with keying material.

### 4. Security Considerations

This entire document is about security.

### 5. IANA Considerations

This document does not require any actions by IANA.

### 6. Acknowledgements

The author would like to thank Stefanie Gerdes for her review comments.

### 7. Informative References

[I-D.ietf-abfab-arch]

Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-10 (work in progress), December 2013.

[I-D.ietf-kitten-sasl-oauth]

Mills, W., Showalter, T., and H. Tschofenig, "A set of SASL Mechanisms for OAuth", draft-ietf-kitten-sasl-oauth-12 (work in progress), December 2013.

- [I-D.ietf-lwig-terminology]  
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-06 (work in progress), December 2013.
- [I-D.ietf-oauth-json-web-token]  
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token-15 (work in progress), January 2014.
- [I-D.ietf-oauth-v2-http-mac]  
Richer, J., Mills, W., Tschofenig, H., and P. Hunt, "OAuth 2.0 Message Authentication Code (MAC) Tokens", draft-ietf-oauth-v2-http-mac-05 (work in progress), January 2014.
- [I-D.nourse-scep]  
Pritikin, M., Nourse, A., and J. Vilhuber, "Simple Certificate Enrollment Protocol", draft-nourse-scep-23 (work in progress), September 2011.
- [I-D.seitz-ace-usecases]  
Seitz, L., Gerdes, S., and G. Selander, "ACE use cases", draft-seitz-ace-usecases-00 (work in progress), February 2014.
- [I-D.tschofenig-oauth-hotk]  
Bradley, J., Hunt, P., Nadalin, A., and H. Tschofenig, "The OAuth 2.0 Authorization Framework: Holder-of-the-Key Token Usage", draft-tschofenig-oauth-hotk-03 (work in progress), January 2014.
- [RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.
- [RFC2704] Blaze, M., Feigenbaum, J., Ioannidis, J., and A. Keromytis, "The KeyNote Trust-Management System Version 2", RFC 2704, September 1999.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.
- [RFC3281] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.

- [RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.
- [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.
- [RFC5934] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Management Protocol (TAMP)", RFC 5934, August 2010.
- [RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, October 2010.
- [RFC6063] Doherty, A., Pei, M., Machani, S., and M. Nystrom, "Dynamic Symmetric Key Provisioning Protocol (DSKPP)", RFC 6063, December 2010.
- [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.
- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.
- [RFC6818] Yee, P., "Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 6818, January 2013.

#### Author's Address

Hannes Tschofenig  
ARM Ltd.  
110 Fulbourn Rd  
Cambridge CB1 9NJ  
Great Britain

Email: [Hannes.tschofenig@gmx.net](mailto:Hannes.tschofenig@gmx.net)  
URI: <http://www.tschofenig.priv.at>