OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 28, 2018

B. Campbell
Ping Identity
J. Bradley
Yubico
N. Sakimura
Nomura Research Institute
T. Lodderstedt
YES Europe AG

July 27, 2017

Mutual TLS Profile for OAuth 2.0
draft-ietf-oauth-mtls-03

Abstract

This document describes Transport Layer Security (TLS) mutual authentication using X.509 certificates as a mechanism for OAuth client authentication to the token endpoint as well as for certificate bound sender constrained access tokens.

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 28, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document describes Transport Layer Security (TLS) mutual authentication using X.509 certificates as a mechanism for OAuth client authentication to the token endpoint as well as for sender constrained access to OAuth protected resources.

The OAuth 2.0 Authorization Framework [RFC6749] defines a shared secret method of client authentication but also allows for the definition and use of additional client authentication mechanisms when interacting with the authorization server's token endpoint. This document describes an additional mechanism of client authentication utilizing mutual TLS [RFC5246] certificate-based authentication, which provides better security characteristics than shared secrets.

Mutual TLS sender constrained access to protected resources ensures that only the party in possession of the private key corresponding to the certificate can utilize the access token to get access to the associated resources. Such a constraint is unlike the case of the basic bearer token described in [RFC6750], where any party in possession of the access token can use it to access the associated resources. Mutual TLS sender constrained access binds the access token to the client's certificate thus preventing the use of stolen access tokens or replay of access tokens by unauthorized parties.

Mutual TLS sender constrained access tokens and mutual TLS client authentication are distinct mechanisms that don't necessarily need to be deployed together.

1.1. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology

This specification uses the following phrases interchangeably:

Transport Layer Security (TLS) Mutual Authentication

Mutual TLS

These phrases all refer to the process whereby a client uses it's X.509 certificate to authenticate itself with a server when negotiating a TLS session. In TLS 1.2 [RFC5246] this requires the

client to send Client Certificate and Certificate Verify messages during the TLS handshake and for the server to verify these messages.

2. Mutual TLS for Client Authentication

2.1. Mutual TLS Client Authentication to the Token Endpoint

The following section defines, as an extension of OAuth 2.0, Section 2.3 [RFC6749], the use of mutual TLS X.509 client certificates as client credentials. The requirement of mutual TLS for client authentications is determined by the authorization server based on policy or configuration for the given client (regardless of whether the client was dynamically registered or statically configured or otherwise established). OAuth 2.0 requires that access token requests by the client to the token endpoint use TLS. In order to utilize TLS for client authentication, the TLS connection MUST have been established or reestablished with mutual X.509 certificate authentication (i.e. the Client Certificate and Certificate Verify messages are sent during the TLS Handshake [RFC5246]).

For all access token requests to the token endpoint, regardless of the grant type used, the client MUST include the "client_id" parameter, described in OAuth 2.0, Section 2.2 [RFC6749]. The presence of the "client_id" parameter enables the authorization server to easily identify the client independently from the content of the certificate and allows for trust models to vary as appropriate for a given deployment. The authorization server can locate the client configuration by the client identifier and check the certificate presented in the TLS Handshake against the expected credentials for that client. The authorization server MUST enforce some method of binding a certificate to a client. The following two binding methods are defined:

PKI The PKI method uses a distinguished name (DN) to identify the client. The TLS handshake is utilized to validate the client's possession of the private key corresponding to the public key in the certificate and to validate the corresponding certificate chain. The client is successfully authenticated if the subject information in the certificate matches the configured DN. The client may prescribe the DN of the issuer of its certificates. The authorization server will enforce this restriction after the TLS handshake took place. Setting the issuer to a certain CA securely scopes the DN of the client to this CA and shall prevent an attacker from impersonating a client by using a certificate for the client's DN obtained from a different CA. The PKI method facilitates the way X.509 certificates are traditionally being used for authentication. It also allows the client to rotate its

X.509 certificates without the need to modify its respective authentication data at the authorization server.

Public Key The Public Key method uses public keys to identify clients. As pre-requisite, the client registers a X.509 certificate or a trusted source for its X.509 certificates (jwks uri as defined in [RFC7591]) with the authorization server. During authentication, TLS is utilized to validate the client's possession of the private key corresponding to the public key presented in the respective TLS handshake. In contrast to the PKI method, the certificate chain is not validated in this case. The client is successfully authenticated, if the subject public key info of the validated certificate matches the subject public key info of one the certificates configured for that particular client. The Public Key method allows to use mutual TLS to authenticate clients without the need to maintain a PKI. When used in conjunction with a trusted X.509 certificate source, it also allows the client to rotate its X.509 certificates without the need to change its respective authentication data at the authorization server.

2.2. Authorization Server Metadata

In authorization server metadata, such as [OpenID.Discovery] and [I-D.ietf-oauth-discovery], the "token_endpoint_auth_methods_supported" parameter indicates client authentication methods to the token endpoint supported by the authorization server. This document introduces the value "tls_client_auth" for use in "token_endpoint_auth_methods_supported" to indicate server support for mutual TLS client authentication utilizing the PKI method. And for the support of mutual TLS client authentication utilizing the Public Key method, the value "pub_key_tls_client_auth" is used in "token_endpoint_auth_methods_supported".

This document also introduces a new authorization server metadata parameter:

mutual_tls_sender_constrained_access_tokens
OPTIONAL. Boolean value indicating server support for mutual TLS sender constrained access tokens. If omitted, the default value is "false".

2.3. Dynamic Client Registration

This document adds the following values and metadata parameters to OAuth 2.0 Dynamic Client Registration [RFC7591].

The client metadata parameter "mutual_tls_sender_constrained_access_tokens" is a Boolean value used to indicate the client's intention to use mutual TLS sender constrained access tokens. If omitted, the default value is "false".

For the PKI method of binding a certificate to a client, the value "tls_client_auth" is used to indicate the client's intention to use mutual TLS as an authentication method to the token endpoint for the "token_endpoint_auth_method" client metadata parameter. And the following two metadata parameters are introduced in support of the PKI method of binding a certificate to a client:

tls_client_auth_subject_dn
: An [RFC4514] string representation of the expected subject distinguished name of the certificate the OAuth client will use in mutual TLS authentication.

tls_client_auth_root_dn
: OPTIONAL. An [RFC4514] string representation of a distinguished name that can optionally be used to constrain, for the given client, the expected distinguished name of the root issuer of the client certificate.

With the Public Key method of binding a certificate to a client, the value "pub_key_tls_client_auth" is used for the "token_endpoint_auth_method" client metadata parameter to indicate the client's intention to use mutual TLS with a self-signed certificate as an authentication method. For the Public Key method, the existing "jwks_uri" or "jwks" metadata parameters from [RFC7591] are used to convey client's public keys, where the X.509 certificates are represented using the "x5c" parameter from [RFC7517].

3. Mutual TLS Sender Constrained Resources Access

When mutual TLS is used at the token endpoint, the authorization server is able to bind the issued access token to the client certificate. Such a binding is accomplished by associating the certificate with the token in a way that can be accessed by the protected resource, such as embedding the certificate hash in the issued access token directly, using the syntax described in Section 3.1, or through token introspection as described in Section 3.2. Other methods of associating a certificate with an access token are possible, per agreement by the authorization server and the protected resource, but are beyond the scope of this specification.

The client makes protected resource requests as described in [RFC6750], however, those requests MUST be made over a mutually

authenticated TLS connection using the same certificate that was used for mutual TLS at the token endpoint.

The protected resource MUST obtain the client certificate used for mutual TLS authentication and MUST verify that the certificate matches the certificate associated with the access token. If they do not match, the resource access attempt MUST be rejected with an error per [RFC6750] using an HTTP 401 status code and the "invalid_token" error code.

3.1. X.509 Certificate SHA-256 Thumbprint Confirmation Method for JWT

When access tokens are represented as a JSON Web Tokens (JWT)[RFC7519], the certificate hash information SHOULD be represented using the "x5t#S256" confirmation method member defined herein.

To represent the hash of a certificate in a JWT, this specification defines the new JWT Confirmation Method RFC 7800 [RFC7800] member "x5t#S256" for the X.509 Certificate SHA-256 Thumbprint. The value of the "x5t#S256" member is a base64url-encoded SHA-256[SHS] hash (a.k.a. thumbprint or digest) of the DER encoding of the X.509 certificate[RFC5280] (note that certificate thumbprints are also sometimes also known as certificate fingerprints).

The following is an example of a JWT payload containing an "x5t#S256" certificate thumbprint confirmation method.

```
{
  "iss": "https://server.example.com",
  "sub": "ty.webb@example.com",
  "exp": 1493726400,
  "nbf": 1493722800,
  "cnf":{
    "x5t#S256": "bwcK0esc3ACC3DB2Y5_lESsXE8o9ltc05O89jdN-dg2"
  }
}
```

Figure 1: Example claims of a Certificate Thumbprint Constrained JWT

If, in the future, certificate thumbprints need to be computed using hash functions other than SHA-256, it is suggested that additional related JWT confirmation methods members be defined for that purpose. For example, a new "x5t#S512" (X.509 Certificate Thumbprint using SHA-512) confirmation method member could be defined by registering it in the the IANA "JWT Confirmation Methods" registry [IANA.JWT.Claims] for JWT "cnf" member values established by [RFC7800].

3.2. Confirmation Method for Token Introspection

OAuth 2.0 Token Introspection [RFC7662] defines a method for a protected resource to query an authorization server about the active state of an access token as well as to determine meta-information about the token.

For a mutual TLS sender constrained access token, the hash of the certificate to which the token is bound is conveyed to the protected resource as meta-information in a token introspection response. The hash is conveyed using same structure as the certificate SHA-256 thumbprint confirmation method, described in Section 3.1, as a top-level member of the introspection response JSON. The protected resource compares that certificate hash to a hash of the client certificate used for mutual TLS authentication and rejects the request, if they do not match.

Proof-of-Possession Key Semantics for JSON Web Tokens [RFC7800] defined the "cnf" (confirmation) claim, which enables confirmation key information to be carried in a JWT. However, the same proof-of-possession semantics are also useful for introspected access tokens whereby the protected resource obtains the confirmation key data as meta-information of a token introspection response and uses that information in verifying proof-of-possession. Therefore this specification defines and registers proof-of-possession semantics for OAuth 2.0 Token Introspection [RFC7662] using the "cnf" structure. When included as a top-level member of an OAuth token introspection response, "cnf" has the same semantics and format as the claim of the same name defined in [RFC7800]. While this specification only explicitly uses the "x5t#S256" confirmation method member, it needed to define and register the higher level "cnf" structure as an introspection response member in order to define and use its more specific "x5t#S256" confirmation method.

The following is an example of an introspection response for an active token with an "x5t#S256" certificate thumbprint confirmation method.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "active": true,
  "iss": "https://server.example.com",
  "sub": "ty.webb@example.com",
  "exp": 1493726400,
  "nbf": 1493722800,
  "cnf":{
    "x5t#S256": "bwcK0esc3ACC3DB2Y5_lESsXE8o9ltc05O89jdN-dg2"
  }
}
```

Figure 2: Example Introspection Response for a Certificate Constrained Access Token

4. Implementation Considerations

4.1. Authorization Server

The authorization server needs to setup its TLS configuration appropriately for the binding methods it supports.

If the authorization server wants to support mutual TLS client authentication and other client authentication methods in parallel, it should make mutual TLS optional on the token endpoint.

If the authorization server supports the Public Key method, it should configure the TLS stack in a way that it does not verify whether the certificate presented by the client during the handshake is signed by a trusted CA certificate.

Please note: the Public Key method is intended to support client authentication using self-signed certificates.

The authorization server may also consider hosting the token endpoint on a separate host name in order to prevent unintended impact on the TLS behavior of its other endpoints, e.g. authorization or registration.

4.2. Resource Server

From the perspective of the resource server, TLS client authentication is used as a proof of possession method only. For the purpose of client authentication, the resource server may completely rely on the authorization server. So there is no need to validate the trust chain of the client's certificate in any of the methods

defined in this document. The resource server should therefore configure the TLS stack in a way that it does not verify whether the certificate presented by the client during the handshake is signed by a trusted CA certificate.

4.3. Sender Constrained Access Tokens Without Client Authentication

This document allows for the use of client authentication only or client authentication in combination with sender constraint access tokens. Use of mutual TLS sender constrained access tokens without client authentication (e.g. to support binding access tokens to a TLS client certificate for public clients) is also possible. The authorization server would configure the TLS stack in the same manor as for the Public Key method such that it does not verify that the certificate presented by the client during the handshake is signed by a trusted CA. Individual instances of a public client would then create a self-signed certificate for mutual TLS with the authorization server and resource server. The authorization server would not authenticate the client at the OAuth layer but would bind issued access tokens to the certificate, which the client has proven possession of the corresponding private key. The access token is then mutual TLS sender constrained and can only be used by the client possessing the certificate and private key and utilizing them to negotiate mutual TLS on connections to the resource server.

4.4. Certificate Bound Access Tokens

As described in Section 3, an access token is bound to a specific client certificate, which means that the same certificate must be used for mutual TLS on protected resource access. It also implies that access tokens are invalidated when a client updates the certificate, which can be handled similar to expired access tokens where the client requests a new access token (typically with a refresh token) and retries the protected resource request.

5. IANA Considerations

5.1. JWT Confirmation Methods Registration

This specification requests registration of the following value in the IANA "JWT Confirmation Methods" registry [IANA.JWT.Claims] for JWT "cnf" member values established by [RFC7800].

5.1.1. Registry Contents

- Confirmation Method Value: "x5t#S256"
- Confirmation Method Description: X.509 Certificate SHA-256 Thumbprint

- Change Controller: IESG
- Specification Document(s): Section 3.1 of [[this specification]]

5.2. OAuth Authorization Server Metadata Registration

This specification requests registration of the following value in the IANA "OAuth Authorization Server Metadata" registry [IANA.OAuth.Parameters] established by [I-D.ietf-oauth-discovery].

5.2.1. Registry Contents

- Metadata Name: "mutual_tls_sender_constrained_access_tokens"
- Metadata Description: Indicates server support for mutual TLS sender constraint access tokens.
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this specification]]

5.3. Token Endpoint Authentication Method Registration

This specification requests registration of the following value in the IANA "OAuth Token Endpoint Authentication Methods" registry [IANA.OAuth.Parameters] established by [RFC7591].

5.3.1. Registry Contents

- Token Endpoint Authentication Method Name: "tls_client_auth"
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this specification]]

- Token Endpoint Authentication Method Name: "pub_key_tls_client_auth"
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this specification]]

5.4. OAuth Token Introspection Response Registration

This specification requests registration of the following value in the IANA "OAuth Token Introspection Response" registry [IANA.OAuth.Parameters] established by [RFC7662].

5.4.1. Registry Contents

- Claim Name: "cnf"
- Claim Description: Confirmation
- Change Controller: IESG
- Specification Document(s): Section 3.2 of [[this specification]]

5.5. OAuth Dynamic Client Registration Metadata Registration

This specification requests registration of the following client metadata definitions in the IANA "OAuth Dynamic Client Registration Metadata" registry [IANA.OAuth.Parameters] established by [RFC7591]:

5.5.1. Registry Contents

- Client Metadata Name: "mutual_tls_sender_constrained_access_tokens"
- Client Metadata Description: Indicates the client's intention to use mutual TLS sender constraint access tokens.
- Change Controller: IESG
- Specification Document(s): Section 2.3 of [[this specification]]

- Client Metadata Name: "tls_client_auth_subject_dn"
- Client Metadata Description: String value specifying the expected subject distinguished name of the client certificate.
- Change Controller: IESG
- Specification Document(s): Section 2.3 of [[this specification]]

- Client Metadata Name: "tls_client_auth_root_dn"
- Client Metadata Description: String value specifying the expected distinguished name of the root issuer of the client certificate
- Change Controller: IESG
- Specification Document(s): Section 2.3 of [[this specification]]

6. Security Considerations

6.1. TLS Versions and Best Practices

TLS 1.2 [RFC5246] is cited in this document because, at the time of writing, it is latest version that is widely deployed. However, this document is applicable with other TLS versions supporting certificate-based client authentication. Implementation security considerations for TLS, including version recommendations, can be found in Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) [BCP195].

6.2. X.509 Certificate Spoofing

If the PKI method is used, an attacker could try to impersonate a client using a certificate for the same DN issued by another CA, which the authorization server trusts.

There are two ways to cope with that threat: the authorization server may decide to only accept a limited number of CAs whose certificate issuance policy meets its security requirements. Alternatively or in

addition, the client may want to explicitly prescribe the CA it will use for obtaining its certificates. The latter is supported by this document with the client registration parameter "tls_client_auth_root_dn".

7. References

7.1. Normative References

[BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/bcp195>.

[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>.

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, DOI 10.17487/RFC4514, June 2006, <http://www.rfc-editor.org/info/rfc4514>.

[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>.

[RFC5280] 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, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <http://www.rfc-editor.org/info/rfc6749>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <http://www.rfc-editor.org/info/rfc6750>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <http://www.rfc-editor.org/info/rfc7800>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

7.2. Informative References

[I-D.ietf-oauth-discovery]
Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", draft-ietf-oauth-discovery-04 (work in progress), August 2016.

[IANA.JWT.Claims]
IANA, "JSON Web Token Claims", <http://www.iana.org/assignments/jwt>.

[IANA.OAuth.Parameters]
IANA, "OAuth Parameters", <http://www.iana.org/assignments/oauth-parameters>.

[OpenID.Discovery]
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0", February 2014.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <http://www.rfc-editor.org/info/rfc7519>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <http://www.rfc-editor.org/info/rfc7591>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <http://www.rfc-editor.org/info/rfc7662>.

Appendix A. Acknowledgements

Scott "not Tomlinson" Tomilson and Matt Peterson were involved in design and development work on a mutual TLS OAuth client authentication implementation that informed some of the content of this document.

Additionally, the authors would like to thank the following people for their input and contributions to the specification: Sergey Beryozkin, Vladimir Dzhuvinov, Samuel Erdtman, Phil Hunt, Sean Leonard, Kepeng Li, James Manger, Jim Manico, Nov Matake, Sascha Preibisch, Justin Richer, Dave Tonge, and Hannes Tschofenig.

Appendix B. Document(s) History

[[to be removed by the RFC Editor before publication as an RFC]]

draft-ietf-oauth-mtls-03

- Introduced metadata and client registration parameter to publish and request support for mutual TLS sender constrained access tokens
- Added description of two methods of binding the cert and client, PKI and Public Key.
- Indicated that the "tls_client_auth" authentication method is for the PKI method and introduced "pub_key_tls_client_auth" for the Public Key method
- Added implementation considerations, mainly regarding TLS stack configuration and trust chain validation, as well as how to to do binding of access tokens to a TLS client certificate for public clients, and considerations around certificate bound access tokens
- Added new section to security considerations on cert spoofing
- Add text suggesting that a new cnf member be defined in the future, if hash function(s) other than SHA-256 need to be used for certificate thumbprints

draft-ietf-oauth-mtls-02

- Fixed editorial issue https://mailarchive.ietf.org/arch/msg/oauth/U46UMEh8XIOQnvXY9pHFq1MKPns
- Changed the title (hopefully "Mutual TLS Profile for OAuth 2.0" is better than "Mutual TLS Profiles for OAuth Clients").

draft-ietf-oauth-mtls-01

- Added more explicit details of using RFC 7662 token introspection with mutual TLS sender constrained access tokens.

- Added an IANA OAuth Token Introspection Response Registration request for "cnf".
- Specify that tls_client_auth_subject_dn and tls_client_auth_root_dn are RFC 4514 String Representation of Distinguished Names.
- Changed tls_client_auth_issuer_dn to tls_client_auth_root_dn.
- Changed the text in the Section 3 to not be specific about using a hash of the cert.
- Changed the abbreviated title to 'OAuth Mutual TLS' (previously was the acronym MTLSPOC).

draft-ietf-oauth-mtls-00

- Created the initial working group version from draft-campbell-oauth-mtls

draft-campbell-oauth-mtls-01

- Fix some typos.
- Add to the acknowledgements list.

draft-campbell-oauth-mtls-00

- Add a Mutual TLS sender constrained protected resource access method and a x5t#S256 cnf method for JWT access tokens (concepts taken in part from draft-sakimura-oauth-jpop-04).
- Fixed "token_endpoint_auth_methods_supported" to "token_endpoint_auth_method" for client metadata.
- Add "tls_client_auth_subject_dn" and "tls_client_auth_issuer_dn" client metadata parameters and mention using "jwks_uri" or "jwks".
- Say that the authentication method is determined by client policy regardless of whether the client was dynamically registered or statically configured.
- Expand acknowledgements to those that participated in discussions around draft-campbell-oauth-tls-client-auth-00
- Add Nat Sakimura and Torsten Lodderstedt to the author list.

draft-campbell-oauth-tls-client-auth-00

- Initial draft.

Authors' Addresses

Brian Campbell
Ping Identity

Email: brian.d.campbell@gmail.com

John Bradley
Yubico

Email: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/

Nat Sakimura
Nomura Research Institute

Email: n-sakimura@nri.co.jp
URI: https://nat.sakimura.org/

Torsten Lodderstedt
YES Europe AG

Email: torsten@lodderstedt.net