Workgroup: Aspirational
Internet-Draft:
draft-bdc-something-something-certificate-05
Published: 23 March 2021
Intended Status: Informational
Expires: 24 September 2021
Authors: B. Campbell
Ping Identity

Client-Cert HTTP Header: Conveying Client Certificate Information from TLS Terminating Reverse Proxies to Origin Server Applications

Abstract

This document defines the HTTP header field Client-Cert that allows a TLS terminating reverse proxy to convey the client certificate of a mutually-authenticated TLS connection to the origin server in a common and predictable manner.

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 https://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 24 September 2021.

Copyright Notice

Copyright (c) 2021 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 (https://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

A fairly common deployment pattern for HTTPS applications is to have the origin HTTP application servers sit behind a reverse proxy that terminates TLS connections from clients. The proxy is accessible to the internet and dispatches client requests to the appropriate origin server within a private or protected network. The origin servers are not directly accessible by clients and are only reachable through the reverse proxy. The backend details of this type of deployment are typically opaque to clients who make requests to the proxy server and see responses as though they originated from the proxy server itself. Although HTTPS is also usually employed between the proxy and the origin server, the TLS connection that the client establishes for HTTPS is only between itself and the reverse proxy server.

The deployment pattern is found in a number of varieties such as n-tier architectures, content delivery networks, application load balancing services, and ingress controllers.

Although not exceedingly prevalent, TLS client certificate authentication is sometimes employed and in such cases the origin server often requires information about the client certificate for its application logic. Such logic might include access control decisions, audit logging, and binding issued tokens or cookies to a certificate, and the respective validation of such bindings. The specific details from the certificate needed also vary with the application requirements. In order for these types of application deployments to work in practice, the reverse proxy needs to convey information about the client certificate to the origin application server. A common way this information is conveyed in practice today is by using non-standard headers to carry the certificate (in some encoding) or individual parts thereof in the HTTP request that is dispatched to the origin server. This solution works but interoperability between independently developed components can be cumbersome or even impossible depending on the implementation choices respectively made (like what header names are used or are configurable, which parts of the certificate are exposed, or how the certificate is encoded). A well-known predictable approach to this commonly occurring functionality could improve and simplify interoperability between independent implementations.

This document aspires to standardize an HTTP header field named Client-Cert that a TLS terminating reverse proxy (TTRP) adds to requests that it sends to the backend origin servers. The header value contains the client certificate from the mutually-authenticated TLS connection between the originating client and the TTRP. This enables the backend origin server to utilize the client certificate information in its application logic. While there may be additional proxies or hops between the TTRP and the origin server (potentially even with mutually-authenticated TLS connections between them), the scope of the Client-Cert header is intentionally limited to exposing to the origin server the certificate that was presented by the originating client in its connection to the TTRP.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

Phrases like TLS client certificate authentication or mutually-authenticated TLS are used throughout this document to refer to the process whereby, in addition to the normal TLS server authentication with a certificate, a client presents its X.509 certificate [RFC5280] and proves possession of the corresponding private key to a server when negotiating a TLS connection or the resumption of such a connection. In contemporary versions of TLS [RFC8446] [RFC5246] this requires that the client send the Certificate and CertificateVerify messages during the handshake and for the server to verify the CertificateVerify and Finished messages.

[[HTTP2 forbids TLS renegotiation and post-handshake authentication but it's possible with HTTP1.1 and maybe needs to be discussed explicitly here or somewhere in this document? Naively I'd say that the Client-Cert header will be sent with the data of the most recent client cert anytime after renegotiation or post-handshake auth. And only for requests that are fully covered by the cert but that in practice making the determination of where exactly in the application data the cert messages arrived is hard to impossible so it'll be a best effort kind of thing.]]

2. HTTP Header Field and Processing Rules

2.1. Encoding

The field-values of the HTTP header defined herein utilize the following encoded form.

A certificate is represented in text as an EncodedCertificate, which is the base64-encoded (Section 4 of [RFC4648]) DER [ITU.X690] PKIX certificate. The encoded value MUST NOT include any line breaks, whitespace, or other additional characters. ABNF [RFC5234] syntax for EncodedCertificate is shown in the figure below.

```
EncodedCertificate = 1*( DIGIT / ALPHA / "+" / "/" ) 0*2"="

DIGIT = <Defined in Section B.1 of [RFC5234]>  ; A-Z / a-z
ALPHA = <Defined in Section B.1 of [RFC5234]>  ; 0-9
```

2.2. Client-Cert HTTP Header Field

In the context of a TLS terminating reverse proxy (TTRP) deployment, the TTRP makes the TLS client certificate available to the backend application with the following header field.

Client-Cert The end-entity client certificate as an EncodedCertificate value.

The Client-Cert header field defined herein is only for use in HTTP requests and MUST NOT be used in HTTP responses. It is a single HTTP header field-value as defined in Section 3.2 of [RFC7230], which MUST NOT have a list of values or occur multiple times in a request.

2.3. Processing Rules

This section outlines the applicable processing rules for a TLS terminating reverse proxy (TTRP) that has negotiated a mutually-authenticated TLS connection to convey the client certificate from that connection to the backend origin servers. Use of the technique is to be a configuration or deployment option and the processing rules described herein are for servers operating with that option enabled.

A TTRP negotiates the use of a mutually-authenticated TLS connection with the client, such as is described in [RFC8446] or [RFC5246], and validates the client certificate per its policy and trusted certificate authorities. Each HTTP request on the underlying TLS connection are dispatched to the origin server with the following modifications:

1. The client certificate is be placed in the Client-Cert header field of the dispatched request as defined in Section 2.2.

2. Any occurrence of the Client-Cert header in the original incoming request MUST be removed or overwritten before forwarding the request. An incoming request that has a Client-Cert header MAY be rejected with an HTTP 400 response.

Requests made over a TLS connection where the use of client certificate authentication was not negotiated MUST be sanitized by removing any and all occurrences Client-Cert header field prior to dispatching the request to the backend server.

Backend origin servers may then use the Client-Cert header of the request to determine if the connection from the client to the TTRP was mutually-authenticated and, if so, the certificate thereby presented by the client.

Forward proxies and other intermediaries MUST NOT add the Client-Cert header to requests, or modify an existing Client-Cert header. Similarly, clients MUST NOT employ the Client-Cert header in requests.

A server that receives a request with a Client-Cert header value that it considers to be too large can respond with an HTTP 431 status code per Section 5 of [RFC6585].

3. Security Considerations

The header described herein enable a TTRP and backend or origin server to function together as though, from the client's perspective, they are a single logical server side deployment of HTTPS over a mutually-authenticated TLS connection. Use of the Client-Cert header outside that intended use case, however, may undermine the protections afforded by TLS client certificate authentication. Therefore steps MUST be taken to prevent unintended use, both in sending the header and in relying on its value.

Producing and consuming the Client-Cert header SHOULD be a configurable option, respectively, in a TTRP and backend server (or individual application in that server). The default configuration for both should be to not use the Client-Cert header thus requiring an "opt-in" to the functionality.

In order to prevent header injection, backend servers MUST only accept the Client-Cert header from a trusted TTRP (or other proxy in a trusted path from the TTRP). A TTRP MUST sanitize the incoming request before forwarding it on by removing or overwriting any existing instances of the header. Otherwise arbitrary clients can control the header value as seen and used by the backend server. It is important to note that neglecting to prevent header injection does not "fail safe" in that the nominal functionality will still work as expected even when malicious actions are possible. As such, extra care is recommended in ensuring that proper header sanitation is in place.

The communication between a TTRP and backend server needs to be secured against eavesdropping and modification by unintended parties.

The configuration options and request sanitization are necessarily functionally of the respective servers. The other requirements can be met in a number of ways, which will vary based on specific deployments. The communication between a TTRP and backend or origin server, for example, might be authenticated in some way with the insertion and consumption of the Client-Cert header occurring only on that connection. Alternatively the network topology might dictate a private network such that the backend application is only able to accept requests from the TTRP and the proxy can only make requests to that server. Other deployments that meet the requirements set forth herein are also possible.

4. IANA Considerations

[[TBD if this draft progresses, register the Client-Cert HTTP header field in the "Permanent Message Header Field Names" registry defined in [RFC3864]]]

5. Normative References

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[ITU.X690] International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic

Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", August 2015.

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

[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, <https://www.rfc-editor.org/info/rfc5280>.

6. Informative References

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/info/rfc6585>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/info/rfc8705>.

[RFC7239] Petersson, A. and M. Nilsson, "Forwarded HTTP Extension", RFC 7239, DOI 10.17487/RFC7239, June 2014, <https://www.rfc-editor.org/info/rfc7239>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[I-D.ietf-httpbis-header-structure]
Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-19, 3 June 2020, <https://tools.ietf.org/html/draft-ietf-httpbis-header-structure-19>.

[RFC3864]
Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

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

Appendix A. Example

In a hypothetical example where a TLS client presents the client and intermediate certificate from Figure 1 when establishing a mutually-authenticated TLS connection with the TTRP, the proxy would send the Client-Cert header shown in {#example-header} to the backend. Note that line breaks and whitespace have been added to the value of the header field in Figure 2 for display and formatting purposes only.

```
-----BEGIN CERTIFICATE-----
MIIBqDCCAU6gAwIBAgIBBzAKBggqhkjOPQQDAjA6MRswGQYDVQQKDBJMZXQncyBB
dXRoZW50aWNhdGUxGzAZBgNVBAMMEkxBIEludGVybWVkaWF0ZSBDQTAeFw0yMDAx
MTQyMjU1MzNaFw0yMTAxMjMyMjU1MzNaMA0xCzAJBgNVBAMMAkJDMFkwEwYHKoZI
zj0CAQYIKoZIzj0DAQcDQgAE8YnXXfaUgmnMtOXU/IncWalRhebrXmckC8vdgJ1p
5Be5F/3YC8OthxM4+k1M6aEAEFcGzkJiNy6J84y7uzo9M6NyMHAwCQYDVR0TBAIw
ADAfBgNVHSMEGDAWgBRm3WjLa38lbEYCuiCPct0ZaSED2DAOBgNVHQ8BAf8EBAMC
BsAwEwYDVR0lBAwwCgYIKwYBBQUHAwIwHQYDVR0RAQH/BBMwEYEPYmRjQGV4YW1w
bGUuY29tMAoGCCqGSM49BAMCA0gAMEUCIBHda/r1vaL6G3VliL4/Di6YK0Q6bMje
SkC3dFCOOB8TAiEAx/kHSB4urmiZ0NX5r5XarmPk0wmuydBVoU4hBVZ1yhk=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIB5jCCAYugAwIBAgIBFjAKBggqhkjOPQQDAjBWMQswCQYDVQQGEwJVUzEbMBkG
A1UECgwSTGV0J3MgQXV0aGVudGljYXRlMSowKAYDVQQDDCFMZXQncyBBdXRoZW50
aWNhdGUgUm9vdCBBdXRob3JpdHkwHhcNMjAwMTE0MjEzMjMwWhcNMzAwMTExMjEz
MjMwWjA6MRswGQYDVQQKDBJMZXQncyBBdXRoZW50aWNhdGUxGzAZBgNVBAMMEkxB
IEludGVybWVkaWF0ZSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJf+aA54
RC5pyLAR5yfXVYmNpgd+CGUTDp2KOGhc0gK91zxhHesEYkdXkpS2UN8Kati+yHtW
CV3kkhCngGyv7RqjZjBkMB0GA1UdDgQWBBRm3WjLa38lbEYCuiCPct0ZaSED2DAf
BgNVHSMEGDAWgBTEA2Q6eecKu9g9yb5glbkhhVINGDASBgNVHRMBAf8ECDAGAQH/
AgEAMA4GA1UdDwEB/wQEAwIBhjAKBggqhkjOPQQDAgNJADBGAiEA5pLvaFwRRkxo
mIAtDIwg9D7gC1xzxBl4r28EzmSO1pcCIQCJUShpSXO9HDIQMUgH69fNDEMHXD3R
RX5gP7kuu2KGMg==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIICBjCCAaygAwIBAgIJAKS0yiqKtlhoMAoGCCqGSM49BAMCMFYxCzAJBgNVBAYT
AlVTMRswGQYDVQQKDBJMZXQncyBBdXRoZW50aWNhdGUxKjAoBgNVBAMMIUxldCdz
IEF1dGhlbnRpY2F0ZSBSb290IEF1dGhvcml0eTAeFw0yMDAxMTQyMTI1NDVaFw00
MDAxMDkyMTI1NDVaMFYxCzAJBgNVBAYTAlVTMRswGQYDVQQKDBJMZXQncyBBdXRo
ZW50aWNhdGUxKjAoBgNVBAMMIUxldCdzIEF1dGhlbnRpY2F0ZSBSb290IEF1dGhv
cml0eTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABFoaHU+Z5bPKmGzlYXtCf+E6
HYj62fORaHDOrt+yyh3H/rTcs7ynFfGn+gyFsrSP3Ez88rajv+U2NfD0o0uZ4Pmj
YzBhMB0GA1UdDgQWBBTEA2Q6eecKu9g9yb5glbkhhVINGDAfBgNVHSMEGDAWgBTE
A2Q6eecKu9g9yb5glbkhhVINGDAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQE
AwIBhjAKBggqhkjOPQQDAgNIADBFAiEAmAeg1ycKHriqHnaD4M/UDBpQRpkmdcRF
YGMg1Qyrkx4CIB4ivz3wQcQkGhcsUZ1SOImd/lq1Q0FLf09rGfLQPWDc
-----END CERTIFICATE-----
```

Figure 1: Certificate Chain (with client certificate first)

```
Client-Cert: MIIBqDCCAU6gAwIBAgIBBzAKBggqhkjOPQQDAjA6MRswGQYDVQQKDBJM
 ZXQncyBBdXRoZW50aWNhdGUxGzAZBgNVBAMMEkxBIEludGVybWVkaWF0ZSBDQTAeFw0y
 MDAxMTQyMjU1MzNaFw0yMTAxMjMyMjU1MzNaMA0xCzAJBgNVBAMMAkJDMFkwEwYHKoZI
 zj0CAQYIKoZIzj0DAQcDQgAE8YnXXfaUgmnMtOXU/IncWalRhebrXmckC8vdgJ1p5Be5
 F/3YC8OthxM4+k1M6aEAEFcGzkJiNy6J84y7uzo9M6NyMHAwCQYDVR0TBAIwADAfBgNV
 HSMEGDAWgBRm3WjLa38lbEYCuiCPct0ZaSED2DAOBgNVHQ8BAf8EBAMCBsAwEwYDVR0l
 BAwwCgYIKwYBBQUHAwIwHQYDVR0RAQH/BBMwEYEPYmRjQGV4YW1wbGUuY29tMAoGCCqG
 SM49BAMCA0gAMEUCIBHda/r1vaL6G3VliL4/Di6YK0Q6bMjeSkC3dFCOOB8TAiEAx/kH
 SB4urmiZ0NX5r5XarmPk0wmuydBVoU4hBVZ1yhk=
```

Figure 2: Header in HTTP Request to Origin Server

Appendix B. Considerations Considered

B.1. Header Injection

This draft requires that the TTRP sanitize the headers of the incoming request by removing or overwriting any existing instances of the Client-Cert header before dispatching that request to the backend application. Otherwise, a client could inject its own Client-Cert header that would appear to the backend to have come from the TTRP. Although numerous other methods of detecting/ preventing header injection are possible; such as the use of a unique secret value as part of the header name or value or the application of a signature, HMAC, or AEAD, there is no common general standardized mechanism. The potential problem of client header injection is not at all unique to the functionality of this draft and it would therefor be inappropriate for this draft to define a one-off solution. In the absence of a generic standardized solution existing currently, stripping/sanitizing the headers is the de facto means of protecting against header injection in practice today. Sanitizing the headers is sufficient when properly implemented and is normative requirement of Section 3.

B.2. The Forwarded HTTP Extension

The Forwarded HTTP header field defined in [RFC7239] allows proxy components to disclose information lost in the proxying process. The TLS client certificate information of concern to this draft could have been communicated with an extension parameter to the Forwarded header field, however, doing so would have had some disadvantages that this draft endeavored to avoid. The Forwarded header syntax allows for information about a full chain of proxied HTTP requests, whereas the Client-Cert header of this document is concerned only with conveying information about the certificate presented by the originating client on the TLS connection to the TTRP (which appears as the server from that client's perspective) to backend applications. The multi-hop syntax of the Forwarded header is expressive but also more complicated, which would make processing it more cumbersome, and more importantly, make properly sanitizing its content as required by Section 3 to prevent header injection considerably more difficult and error prone. Thus, this draft opted for the flatter and more straightforward structure of a single Client-Cert header.

B.3. The Whole Certificate and Only the Whole Certificate

Different applications will have varying requirements about what information from the client certificate is needed, such as the subject and/or issuer distinguished name, subject alternative name(s), serial number, subject public key info, fingerprint, etc.. Furthermore some applications, such as "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens" [RFC8705], make use of the entire certificate. In order to accommodate the latter and ensure wide applicability by not trying to cherry-pick particular certificate information, this draft opted to pass the full encoded certificate as the value of the Client-Cert header.

The handshake and validation of the client certificate (chain) of the mutually-authenticated TLS connection is performed by the TTRP. With the responsibility of certificate validation falling on the TTRP, only the end-entity certificate is passed to the backend - the root Certificate Authority is not included nor are any intermediates.

[[It has been suggested that more information about the certificate chain might be needed/wanted by the backend application (to independently evaluate the cert chain, for example, although that seems like it would be terribly inefficient) and that any intermediates as well as the root should also be somehow conveyed, which is an area for further discussion should this draft progress. One potential approach suggested by a few folks is to allow some configurability in what is sent along with maybe a prefix token to indicate what's being sent - something like Client-Cert: FULL <cert> <intermediate> <anchor> or Client-Cert: EE <cert> as the strawman. Or a perhaps a parameter or other construct of [I-D.ietf-httpbis-header-structure] to indicate what's being sent. It's also been suggested that the end-entity certificate by itself might sometimes be too big (esp. e.g., with some post-quantum signature schemes). Hard to account for it both being too much data and not enough data at the same time. But potentially opening up configuration options to send only specific attribute(s) from the client certificate is a possibility for that. In the author's humble opinion the end-entity certificate by itself strikes a good balance for the vast majority of needs and avoids optionality. But, again, this is an area for further discussion should this draft progress.]]

[[It has also been suggested that maybe considerations for [RFC7250] Raw Public Keys is maybe worth considering. This too is this is an area for further discussion and consideration should this draft progress.]]

Appendix C. Acknowledgements

The author would like to thank the following individuals who've contributed in various ways ranging from just being generally supportive of bringing forth the draft to providing specific feedback or content: Evan Anderson, Annabelle Backman, Mike Bishop,

Rory Hewitt, Fredrik Jeansson, Benjamin Kaduk, Torsten Lodderstedt, Kathleen Moriarty, Mark Nottingham, Mike Ounsworth, Matt Peterson, Eric Rescorla, Justin Richer, Michael Richardson, Joe Salowey, Rich Salz, Mohit Sethi, Rifaat Shekh-Yusef, Travis Spencer, Nick Sullivan, Peter Wu, and Hans Zandbelt.

[[Please let me know if you've been erroneously omitted or if you prefer not to be named]]

Appendix D. Document History

[[To be removed by the RFC Editor before publication as an RFC (should that come to pass)]]

draft-bdc-something-something-certificate-05

* Change intended status of the draft to Informational

* Editorial updates and (hopefully) clarifications

draft-bdc-something-something-certificate-04

* Update reference from draft-ietf-oauth-mtls to RFC8705

draft-bdc-something-something-certificate-03

* Expanded [[further discussion notes]] to capture some of the feedback in and around the presentation of the draft in SECDISPATCH at IETF 107 and add those who've provided such feedback to the acknowledgements

draft-bdc-something-something-certificate-02

* Editorial tweaks + [[further discussion notes]]

draft-bdc-something-something-certificate-01

* Use the RFC v3 Format or die trying

draft-bdc-something-something-certificate-00

* Initial draft after a time constrained and rushed secdispatch presentation at IETF 106 in Singapore with the recommendation to write up a draft (at the end of the minutes) and some folks expressing interest despite the rather poor presentation

Author's Address

Brian Campbell
Ping Identity

Email: bcampbell@pingidentity.com