

Universal Serial Bus Type-C Authentication Specification

**Revision 1.0
March 25, 2016**

Copyright © 2016, USB 3.0 Promoter Group
All rights reserved.

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. THE PROVISION OF THIS SPECIFICATION TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

All product names are trademarks, registered trademarks, or service marks of their respective owners.

CONTENTS

Specification Work Group Chairs / Specification Editors	7
Specification Work Group Contributors	7
1 Introduction	10
1.1 Scope.....	10
1.2 Overview	10
1.3 Related Documents.....	10
1.4 Terms and Abbreviations	12
1.5 Conventions.....	13
1.5.1 Precedence	13
1.5.2 Keywords	13
1.5.3 Numbering.....	14
1.5.4 Byte Ordering	14
2 Overview	15
2.1 Topology	15
2.2 Cryptographic Methods.....	15
2.2.1 Random Numbers	16
2.3 Security Overview	16
2.3.1 Periodic Re-Authentication	16
2.3.2 Secret Key Storage and Protection.....	16
2.3.3 Security Evaluation Criteria.....	16
2.4 Impact to Existing Ecosystem	17
2.4.1 Proxy Capabilities (PD traversing the Hub topology)	17
3 Authentication Architecture	18
3.1 Certificates.....	18
3.1.1 Format	18
3.1.2 Textual Format.....	18
3.1.3 Attributes and Extensions	18
3.2 Certificate Chains.....	20
3.2.1 Provisioning.....	20
3.3 Private Keys.....	20
4 Authentication Protocol	21
4.1 Digest Query	21
4.2 Certificate Chain Read	21
4.3 Authentication Challenge	21
4.4 Errors and Alerts	21
4.4.1 Invalid Request	21
4.4.2 Unsupported Protocol Version.....	22
4.4.3 Busy	22
4.4.4 Unspecified	22
5 Authentication Messages.....	23
5.1 Header.....	23

5.1.1	USB Type-C Authentication Protocol Version.....	23
5.1.2	Message Type.....	23
5.1.3	Param1.....	23
5.1.4	Param2.....	24
5.2	Authentication Requests	24
5.2.1	GET_DIGESTS	24
5.2.2	GET_CERTIFICATE.....	24
5.2.3	CHALLENGE.....	25
5.3	Authentication Responses.....	25
5.3.1	DIGESTS.....	26
5.3.2	CERTIFICATE	26
5.3.3	CHALLENGE_AUTH.....	27
5.3.4	ERROR.....	28
6	Authentication of PD Products	29
6.1	Transfers less than or equal to <i>MaxExtendedMsgLen</i>	29
6.2	Transfers greater than <i>MaxExtendedMsgLen</i>	29
6.3	Examples	30
6.4	Timing Requirements for PD Security Extended Messages	32
6.4.1	Authentication Initiator	32
6.4.2	Authentication Responder	32
6.5	Context Hash	33
7	Authentication of USB Products.....	34
7.1	Descriptors	34
7.1.1	Authentication Capability Descriptor	34
7.2	Mapping Authentication Messages to USB	34
7.2.1	Authentication IN	35
7.2.2	Authentication OUT	35
7.3	Authentication Protocol.....	36
7.3.1	Digest Query.....	36
7.3.2	Certificate Read.....	36
7.3.3	Authentication Challenge.....	37
7.3.4	Errors	37
7.4	Timing Requirements for USB	38
7.5	Context Hash	39
8	Protocol Constants	40
A	ACD.....	41
A.1.	ACD Formatting.....	41
A.1.1.	Version TLV.....	41
A.1.2.	XID TLV	42
A.1.3.	Power Source Capabilities TLV	42
A.1.4.	Power Source Certifications TLV	43
A.1.5.	Cable Capabilities TLV	44
A.1.6.	Security Description TLV	44

A.1.7.	Playpen TLV	49
A.1.8.	Vendor Extension TLV	50
A.1.9.	Extension TLV	50
A.2.	ACD for a PD Product	50
A.3.	ACD for a USB Product	51
B	Sample Certificate Chain	52
B.1.	Root Certificate	52
B.2.	First Intermediate Certificate	53
B.3.	Second Intermediate Certificate	54
B.4.	Leaf Certificate	55
C	Potential Attack Vectors	56

TABLES

Table 1-1: Terms and Abbreviations	12
Table 2-1: Summary of Cryptographic Methods	16
Table 3-1: Certificate Chain Format	20
Table 5-1: Authentication Message Header	23
Table 5-2: USB Type-C Authentication Protocol Version	23
Table 5-3: Authentication Request Types	24
Table 5-4: GET_DIGESTS Request Header	24
Table 5-5: GET_CERTIFICATE Request Header	24
Table 5-6: GET_CERTIFICATE Request Payload	25
Table 5-7: CHALLENGE Request Header	25
Table 5-8: CHALLENGE Request Payload	25
Table 5-9: Authentication Response Types	25
Table 5-10: DIGESTS Response Header	26
Table 5-11: DIGESTS Response Payload	26
Table 5-12: CERTIFICATE Response Header	26
Table 5-13: CERTIFICATE Response Payload	26
Table 5-14: CHALLENGE_AUTH Response Header	27
Table 5-15: CHALLENGE_AUTH Response Payload	27
Table 5-16: Message Contents for ECDSA Digital Signature	27
Table 5-17: ERROR Response Header	28
Table 5-18: ERROR Codes	28
Table 6-1: Timeout Values for a PD Authentication Initiator	32
Table 6-2: Timing Requirements for PD Authentication Responder	33
Table 7-1: Authentication Capability Descriptor	34
Table 7-2: Authentication Capability Descriptor Types	34
Table 7-3: Authentication Message <i>bRequest</i> Values	35
Table 7-4: Authentication IN Control Request Fields	35
Table 7-5: Authentication Message Header Mapping	35
Table 7-6: Authentication OUT Control Request Fields	36
Table 7-7: GET_DIGESTS Authentication IN Control Request Fields	36

Table 7-8: GET_CERTIFICATE Authentication OUT Control Request Fields	36
Table 7-9: CERTIFICATE Authentication IN Control Request Fields.....	37
Table 7-10: CHALLENGE Authentication OUT Control Request Fields	37
Table 7-11: CHALLENGE_AUTH Authentication IN Control Request Fields	37
Table 7-12: Authentication Initiator Timeout Values	38
Table 8-1: Protocol Constants.....	40
Table A-1: TLV General Format.....	41
Table A-2: TLV Types.....	41
Table A-3: Version TLV Fields	41
Table A-4: ACD Version Encoding.....	42
Table A-5: XID TLV Fields	42
Table A-6: Power Source Capabilities TLV Fields	42
Table A-7: Power Source Capabilities TLV Data	43
Table A-8: Power Source Certifications TLV Fields.....	44
Table A-9: Cable Capabilities TLV Fields.....	44
Table A-10: Cable Capabilities TLV Data.....	44
Table A-11: Security Description TLV Fields	44
Table A-12: Security Data	45
Table A-13: FIPS/ISO Level Identifiers	45
Table A-14: Vulnerability Assessment.....	46
Table A-15: EAL Encodings	47
Table A-16: Protection Profile Encoding	47
Table A-17: Development Security	48
Table A-18: Certification Maintenance	48
Table A-19: Testing Method Encoding.....	49
Table A-20: Vulnerability Assessment.....	49
Table A-21: Playpen TLV Fields	50
Table A-22: Vendor Extension TLV Fields.....	50
Table A-23: Extension TLV Fields	50
Table A-24: PD Product ACD TLVs.....	51
Table A-25: USB Product ACD TLVs	51

FIGURES

Figure 2-1 Sample Topology.....	15
Figure 6-1 Example Security Transfer Process for an Authentication Initiator	30
Figure 6-2 Example Security Transfer Process for an Authentication Responder	31
Figure 6-3 Example 612-Byte Certificate Chain Read	31
Figure A-1: Bitmap of Version TLV Data	42
Figure A-2: Bitmap of the Common Criteria Identifier	46
Figure A-3: Bitmap of the Security Analysis Identifier	48

Specification Work Group Chairs / Specification Editors

Intel Corporation	Co-Chairs	Bob Dunstan	Abdul Ismail
	Editor	Stephanie Wallick	

Specification Work Group Contributors

Advanced Micro Devices	Jason Hawken	Joseph Scanlon	
Apple	Colin Whitby-Strevens	Robert Walsh	Reese Schreiber
	David Conroy	David Sekowski	
Atmel Corporation	Kerry Maletsky	Stephen Clark	Michel Guellec
	Ronald Ih		
Cypress Semiconductor	Subu Sankaran	Jagadeesan Raj	Anup Nayak
	Jan-Willem van der Waert		
Dell Inc.	Sean O'Neal	Mohammed Hijazi	Frank Molsberry
	Dan Hamlin	Rick Martinez	
DisplayLink (UK) Ltd.	Richard Petrie	Pete Burgers	Dan Ellis
Fresco Logic Inc.	Bob McVay	Tom Burton	Christopher Meyers
	Thomas Huang		
Google Inc.	Adam Langley	William Richardson	Adam Rodriguez
	David Schneider	Mark Hayter	Ken Wu
	Will Drewry	Jerry Parson	Sanjay Krishnan
HP Inc.	Alan Berkema	Jim Waldron	Daniel Hong
Infineon Technologies	Thomas Poeppelmann	Wolfgang Furtner	Harald Hewel
	Wieland Fischer	Sie Boo Chiang	
Intel Corporation	Brad Saunders	David Johnston	Chia-Hung Kuo
	Christine Krause	Rolf Kuhn	Steve McGowan
	Andrew Reinders	Purushottam Goel	Karthi Vadivelu
Lattice Semiconductor	Hoon Choi	Thomas Watzka	
MCCI Corporation	Terry Moore		
Microchip Technology Inc.	Richard Wahler	Mark Bohm	Atish Ghosh
	Robert Schoepflin		
Microsoft Corporation	Niels Ferguson	Nathan Sherman	Martin Borge
	Kinshumann Kinshumann	Vivek Gupta	Toby Nixon
	Kai Inha	Robbie Harris	Andrea Keating
	Fred Bhesania	Jayson Kastens	Rahul Ramadas
NXP Semiconductors	Vijendra Kuroodi	Joe Salvador	Alicia da Conceição
	Krishnan TN		
Renesas Electronics Corp.	Philip Leung	Hideyuki Tanaka	Yuji Asano
	Kentaro Omata	Yoshiyuki Tomoda	Kiichi Muto
	Masahiko Nagata	Chizuru Matsunaga	Toshifumi Yamaoka
ROHM Co., Ltd.	Ruben Balbuena	Kris Bahar	Nobutaka Itakura
	Takashi Sato		

Samsung Electronics Co., Ltd.	Tong Kim	Jagoun Koo	Soondo Kim
STMicroelectronics	Enrico Gregoratto	Guido Bertoni	Sylvie Wuidart
	Yannick Teglia	Anis Ben-Abdallah	Massimo Panzica
	Andrew Marsh	Joris Delclef	Nathalie Ballot
	Joel Huloux	Bernard Kasser	Dragos Davidescu
	Christophe Lorin		
Synopsys, Inc.	Eric Huang	Morten Christiansen	Gervais Fong
	Venkataraghavan Krishnan	Nivin George	Aaron Yang
	Subramaniam Aravindhan	Bala Babu	Satya Patnala
	Kevin Heilman	John Youn	Zongyao Wen
Texas Instruments	Charles Campbell	Deric Waters	Scott Jackson
Total Phase	Chris Yokum		
VIA Technologies	Terrance Shih	Jay Tseng	Fong-Jim Wang
	Benjamin Pan		

RELEASE 1.0
March 25, 2016

Revision History

Revision	Date	Description
1.0	March 25, 2016	Initial Release

1 Introduction

This specification provides a means for authenticating Products with regard to identification and configuration. Authentication is performed via USB Power Delivery message communications and/or via USB data bus control transactions.

USB Type-C Authentication allows an organization to set and enforce a Policy with regard to acceptable Products. This will permit useful security assurances in real world situations. For example:

- A vendor, concerned about product damage resulting from substandard charging devices, can set a Policy requiring that only certified PD Products be used for charging.
- A user, concerned about charging his phone at a public terminal, can set a Policy in his phone requiring that the phone only charge from certified PD Products.
- An organization, concerned about unidentifiable storage devices gaining access to corporate PC assets, can set a Policy in its PCs requiring that only USB storage devices that have been verified and signed by corporate IT are used.

1.1 Scope

This specification defines the architecture and methodology for unilateral Product Authentication. It is intended to be fully compatible with and extend existing PD and USB infrastructure. Information is provided to allow for Policy enforcement, but individual Policy decisions are not specified.

The Authentication of USB Type-C products that support Alternate Modes is allowed. However, the methods to do so are outside the scope of this specification.

1.2 Overview

This specification provides primitives for unilateral Authentication. The security model defined by this specification permits assurances that a Product is:

- Of a particular type from a particular manufacturer with particular characteristics
- Owned and controlled by a particular organization

Local Policy will determine which features need to be present in an attached Product before accessing or providing a resource (e.g. power, storage, etc.).

Product vendors can add security features beyond those listed in this specification, but the definition and implementation of those features is up to the vendor. Added features cannot alter the base specifications defined herein.

1.3 Related Documents

- **USB2.0** – Universal Serial Bus Specification, Revision 2.0, (including errata and ECNs through August 11, 2014) (referred to in this document as the USB 2.0 Specification) (available at: <http://www.usb.org/developers/docs.>)
- **USB3.1** – Universal Serial Bus 3.1 Specification, Revision 1.0, (including errata and ECNs through August 11, 2014) (referred to in this document as the USB 3.1 Specification) (available at: <http://www.usb.org/developers/docs.>)
- **USBPD** – Universal Serial Bus Power Delivery Specification, Revision 3, Version 1.0a, March 25, 2016 (referred to in this document as the USB PD Specification) (available at: <http://www.usb.org/developers/docs.>)

- **USBTYPEC** – Universal Serial Bus Type-C Cable and Connector Specification, Revision 1.2, March 25, 2016 (referred to in this document as the USB Type-C Specification)(available at: <http://www.usb.org/developers/docs>.)
- **USBTYPEC BRIDGE** Universal Serial Bus Type-C Bridge Specification, Revision 1.0, March 25, 2016, (available at <http://www.usb.org/developers/docs>.)
- **ASN.1** - ISO-822-1-4;
 - ITU-T X.680 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.680-201508-I!!PDF-E&type=items);
 - ITU-T X.681 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.681-201508-I!!PDF-E&type=items);
 - ITU-T X.682 (Available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.682-201508-I!!PDF-E&type=items);
 - ITU-T X.683 (Available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.683-201508-I!!PDF-E&type=items.)
- **DER** - ISO-8825-1; ITU-T X.690 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.690-201508-I!!PDF-E&type=items.)
- **X509v3** - ISO-9594-8; ITU-T X.509 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-201210-I!!PDF-E&type=items.)
- **Common Criteria** – Common Criteria for Information Technology Security Evaluation, Parts 1-3, Version 3.1, Revision 4, September 2010 (available at: <https://www.commoncriteriaportal.org/cc/#supporting>)
- **ECDSA** - ANSI X9.62; NIST-FIPS-186-4, Section 6 (available at: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.)
- **NIST P256, secp256r1** - Certicom-SEC-2 (available at: <http://www.secg.org/sec2-v2.pdf>); NIST-Recommended-EC (available at: <http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf>.)
- **SHA256** - NIST-FIPS-180-4 (available at: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.)
- **OID** - ITU-T X.402 (available at: <https://www.itu.int/rec/T-REC-X.402-199906-I/en>.)
- **SP800-90A** – NIST-SP-800-90A (available at: <http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf>.)
- **SP800-90B** – NIST-SP-800-90B (available at: http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf.)¹

¹ Note that this document has not been finalized.

² Unless specified otherwise, all standards specified, including those from ISO, ITU, and NIST, refer to the version or edition which is more recent, as of 1 January 2016.

1.4 Terms and Abbreviations

This section defines the terms and abbreviations used throughout this document.

Table 1-1: Terms and Abbreviations

Term/Abbreviation	Definition
ACD	Additional Certificate Data for a Product.
Authentication	The process of determining whether an Entity is in fact who or what it claims to be.
Authentication Initiator	Refers to a Product that initiates Authentication.
Authentication Responder	Refers to a Product with whom an Authentication Initiator is attempting to authenticate.
Certificate (Cert)	A digital form of identification that provides information about an Entity and certifies ownership of a particular public key.
Certificate Authority (CA)	An Entity that issues Certificates.
Certificate Chain	A series of two or more Certificates where each Certificate is signed by the preceding Certificate in the chain.
Entity	Refers to a Product or an organization, vendor, or manufacturer associated with such Products.
Evaluation Assurance Level (EAL)	The numerical rating describing the depth and rigor of a security evaluation.
Intermediate Certificate	A Certificate that is neither Root nor Leaf.
Leaf Certificate	The last Certificate in a Certificate Chain.
LSB	Least Significant Byte
MSB	Most Significant Byte
Nonce	A number used only once in any given key context. Can be interpreted as N-Once.
OID	Object Identifier. See OID for more details.
PD	USB Power Delivery
PD Product	Source, Sink, or Cable as defined in USBPD
PDUSB Product	A USB Host or USB Device that is capable of both PD and USB communication.
Policy	Policy defines the behavior of Products. It defines the capabilities a Product advertises, its Authentication requirements, and resource availability with respect to unauthenticated Products.
Product	Refers to a PD Product, USB Product, and/or PDUSB Product.

Term/Abbreviation	Definition
Pseudorandom Number Generator (PRNG)	A deterministic algorithm for generating a number or sequence of numbers that are computationally indistinguishable from truly random. See SP800-90A for more details.
Root Certificate	The first Certificate in a Certificate Chain. This certificate is self-signed.
USB Device	A Device (including a Hub) as defined in USB2.0 and USB3.1 .
USB Host	A Host as defined in USB2.0 and USB3.1 .
USB Hub	A Hub as defined in USB2.0 and USB3.1 .
USB Product	A USB Host or USB Device.

1.5 Conventions

1.5.1 Precedence

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, and then text.

1.5.2 Keywords

The following keywords differentiate between the levels of requirements and options.

1.5.2.1 Conditional Normative

Conditional Normative is a keyword used to indicate a feature that is mandatory when another related feature has been implemented. Designers are mandated to implement all such requirements, when the dependent features have been implemented, to ensure interoperability with other compliant Products.

1.5.2.2 Deprecated

Deprecated is a keyword used to indicate a feature, supported in previous releases of the specification, which is no longer supported.

1.5.2.3 Informative

Informative is a keyword that describes information with this specification that intends to discuss and clarify requirements and features as opposed to mandating them.

1.5.2.4 May

May is a keyword that indicates a choice with no implied preference.

1.5.2.5 N/A

N/A is a keyword that indicates that a field or value is not applicable and has no defined value and shall not be checked or used by the recipient.

1.5.2.6 Normative

Normative is a keyword that describes features that are mandated by this specification.

1.5.2.7 Optional/Optionally/Optional Normative

Optional, **Optionally**, and **Optional Normative** are equivalent keywords that describe features not mandated by this specification. However, if an **Optional** feature is implemented, the feature shall be implemented as defined by this specification.

1.5.2.8 Reserved

Reserved is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this specification and, unless otherwise stated, shall not be utilized or adapted by vendor implementation. A **Reserved** bit, byte, word, or field shall be set to zero by the sender and shall be ignored by the receiver. **Reserved** field values shall not be sent by the sender and, if received, shall be ignored by the receiver.

1.5.2.9 Shall/Normative

Shall and **Normative** are keywords indicating a mandatory requirement. Designers are mandated to implement all such requirements to ensure interoperability with other compliant Products.

1.5.2.10 Should

Should is a keyword indicating flexibility of choice with a preferred alternative. Equivalent to the phrase “it is recommended that”.

1.5.3 Numbering

Numbers that are immediately followed by a lowercase “b” (e.g., 01b) are binary values. Numbers that are immediately followed by a lowercase “h” (e.g., 3Ah) are hexadecimal values. Numbers not immediately followed by either a “b”, or “h” are decimal values.

1.5.4 Byte Ordering

The following values in this specification are interpreted as moved over the bus in big-endian order, i.e. MSB to LSB, unless otherwise specified:

- All multiple byte integer values specified with **ASN.1 DER** encoding
- All SHA256 digests

All other multiple byte integer values in this specification are interpreted as and moved over the bus in little-endian order, i.e., LSB to MSB, unless otherwise specified.

The order by which individual bits are moved over a bus is defined in **USBPD** for PD Products and **USB2.0** and **USB3.1** for USB Products.

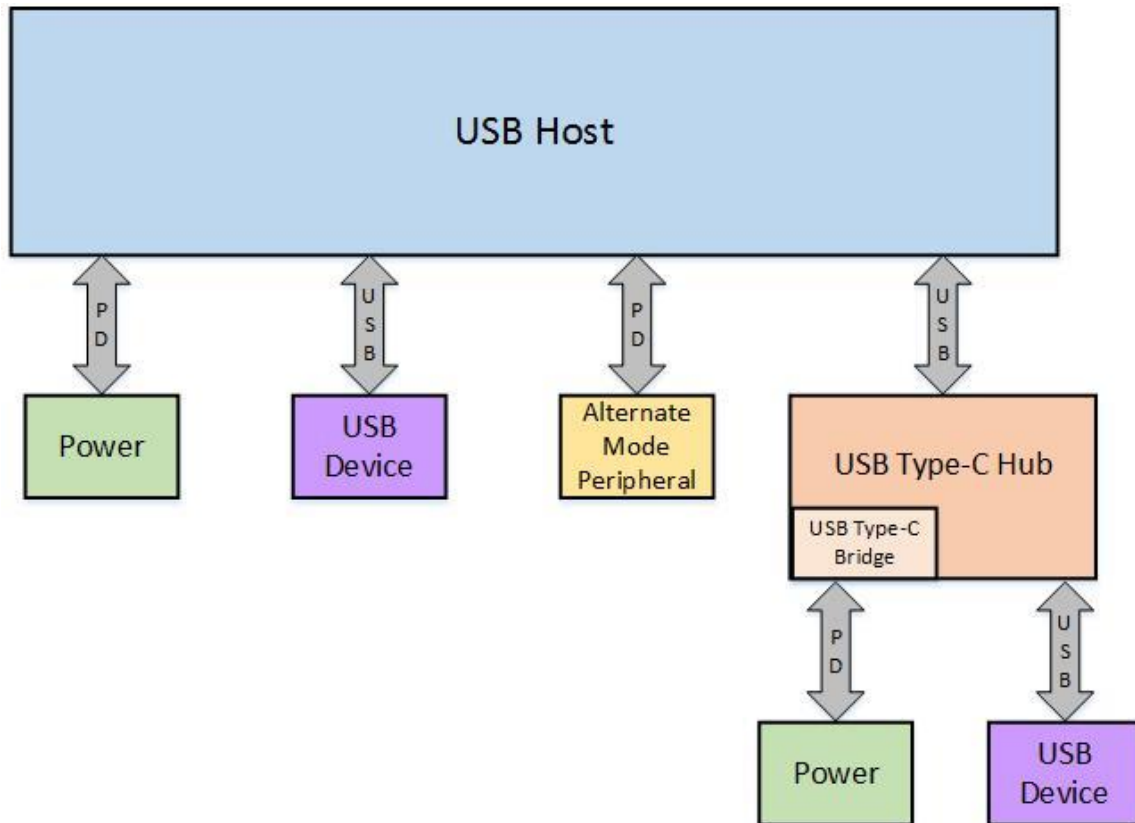
2 Overview

This section contains no Normative requirements.

2.1 Topology

Figure 2-1 represents a sample topology of USB Products and PD Products supporting USB Type-C Authentication. It is not meant to be comprehensive or represent the full range of topology configurations.

Figure 2-1 Sample Topology



2.2 Cryptographic Methods

This specification targets a 128-bit security level for all cryptographic methods. The cryptographic methods used by this specification are shown in Table 2-1.

Table 2-1: Summary of Cryptographic Methods

Method	Use
<i>X509v3</i> , <i>DER</i> encoding	Certificate format
<i>ECDSA</i> using the <i>NIST P256</i> , <i>secp256r1</i> curve, uncompressed point format	Digital signing of Certificates and Authentication Messages
<i>SHA256</i>	Hash algorithm

2.2.1 Random Numbers

The generation of cryptographic keys and the cryptographic protocol exchanges rely on cryptographic quality random numbers. Random numbers are defined as numbers that are distinguishably random by no algorithm with an algorithmic complexity of less than $O(2^{128})$.

The output of a NIST **SP800-90A** compliant PRNG seeded with a 256-bit full **SP800-90B** entropy value is sufficient to meet this standard.

2.3 Security Overview

This specification defines a Certificate-based method for Authentication that allows a Product to authenticate another attached Product and, by Policy, choose how to interact with that Product. For example, a PD Sink may choose not to use the full advertised capabilities of an unauthenticated PD Source. Authentication can be initiated by either a PD Sink, PD Source, or USB Host.

2.3.1 Periodic Re-Authentication

Products can optionally perform periodic re-Authentications. Re-Authentication is used to verify that an authenticated Product has not been replaced by a different Product.

2.3.2 Secret Key Storage and Protection

One threat concerning USB Type-C Authentication is the extraction of secret keys from Products. In a worst case scenario, the extraction of even one secret key could allow an attacker to clone Products in unlimited volume. This or similar scenarios would degrade trust in the whole USB Type-C Authentication ecosystem.

Therefore, it is recommended that vendors of Products take appropriate measures to protect the execution of the USB Type-C Authentication protocol and all private keys. Products should provide protected tamper-resistant operation and storage for the private keys to prevent them from being read (all or in part), copied or otherwise disclosed. This includes protection against side-channel and fault injection attacks, including software exploits and physical attacks such as leakage, probing, glitching, reverse engineering, and statistical analysis methods. Examples of such attacks include Simple and High-Order Differential Power, Electromagnetic, and Fault Analysis attacks. Other examples of attack vectors are listed in Appendix C.

2.3.3 Security Evaluation Criteria

The need for proven and measurable security evaluation results has led to worldwide established certification and evaluation schemes. One of the biggest and most widely applicable security evaluation schemes is **Common Criteria**, which provides global infrastructure for common recognition Certificates. This infrastructure includes government-driven supervision of certification authorities, accredited and capability

balanced evaluation laboratories, and globally harmonized and internationally present mutual recognition contracts. In addition, **Common Criteria** implements a number of different evaluations from which a vendor is free to choose the appropriate level.

It is recommended that the vendor of a Product choose the market-driven individually required level of assurance (EAL), then conduct the independent evaluation process at an accredited evaluation laboratory. After evaluation, the result is published by the certification body in a Certificate which is automatically accepted by a high number of countries.

2.4 Impact to Existing Ecosystem

The impact to existing Products depends largely on individual Policy decisions regarding legacy products (i.e. products that predate this specification). For example, a USB Host with a Policy that allows full functioning of legacy devices will have a minimal impact on the current ecosystem while a USB Host with a Policy that limits or refuses to use functionalities exposed by a legacy product will have a more significant impact.

2.4.1 Proxy Capabilities (PD traversing the Hub topology)

PD is a port-to-port communications protocol. Thus, in order to authenticate PD Products that are connected downstream of a USB Hub, it is necessary to implement a USB Type-C Bridge function in both the USB Hub and corresponding driver in the USB Host. The USB Type-C Bridge is used to translate PD requests to the USB data domain. See **USBTPEC BRIDGE** for more information.

3 Authentication Architecture

3.1 Certificates

3.1.1 Format

All Certificates shall use the **X509v3 ASN.1** structure. All Certificates shall use binary **DER** encoding for **ASN.1**. All Certificates shall use the cryptographic methods listed in Table 2-1. The further description of the Certificate format assumes that the reader is familiar with **X509v3** Certificate terminology.

Leaf certificates shall not exceed *MaxLeafCertSize* in length. Intermediate Certificates shall not exceed *MaxIntermediateCertSize* in length.

3.1.2 Textual Format

All textual **ASN.1** objects contained within Certificates, including DirectoryString, GeneralName, and DisplayText, shall be specified as either a UTF8String, PrintableString, or IA5String. The length of any textual object shall not exceed 64 bytes excluding the **DER** type and **DER** length encoding.

3.1.3 Attributes and Extensions

Where applicable, the Object Identifier (**OID**) is provided.

3.1.3.1 Distinguished Name

The *distinguished name* consists of a number of attributes, which uniquely identify the Entity holding a corresponding private key. A Certificate Authority shall not issue Certificates with the same *distinguished name* to different Entities. *Distinguished name* uniqueness can be accomplished by including an attribute with unique values such as the binary X500UniqueIdentifier or textual *serial number* (See Section 3.1.3.1.3).

3.1.3.1.1 Common Name (OID 2.5.4.3)

This attribute shall appear in every Certificate and shall contain a string matching one of the following three patterns:

- "USB::"
- "USB:<vid>:"
- "USB:<vid>:<pid>"

Where <vid> represents a 4-character lowercase hexadecimal string encoding the 16-bit values corresponding to the VID of the Certificate *subject* and <pid> represents a 4-character lowercase hexadecimal string encoding the 16-bit values corresponding to the PID of the Certificate *subject*. When present, <vid> and <pid> shall be left zero padded and big endian. Uppercase letters shall not be used in the hex encoding of a VID or PID.

The *common name* attribute in the Leaf Certificate of a Certificate Chain shall contain both a VID and a PID. VID and PID are optional in the *common name* attribute of a non-Leaf Certificate. However, if a VID value appears in a Certificate in the Chain, then the same VID value shall be used in all subsequent Certificates. If a PID value appears in a Certificate in the Chain, then the same PID value shall be used in all subsequent Certificates.

3.1.3.1.2 Organization Name (OID 2.5.4.10)

This attribute shall be present in a Root Certificate and may be present in other Certificates. When present, the *organization name* attribute shall contain the human-readable name of the organization that owns the private key that corresponds to the Certificate.

3.1.3.1.3 Serial Number (OID 2.5.4.5)

This attribute is optional. Different PD Products, even from the same production line, have different keys and therefore need distinct *distinguished names*. The *common name* attribute and *organization name* attribute do not provide such uniqueness. There are several attributes inside the *distinguished name* that can be used to make it unique. The recommended method is to use the *serial number* attribute to hold a lowercase hex-encoded value of the binary data (e.g. wafer number, lot number, production lot, etc.) necessary for uniqueness.

3.1.3.2 Basic Constraints (OID 2.5.29.19)

This extension shall be present and marked as critical. The *cA* component shall be false in a Leaf Certificate. The *cA* component shall be true for a non-Leaf Certificate. Other components, including *pathLenConstraint*, shall not be included.

3.1.3.3 Key Usage (OID 2.5.29.15)

This extension shall be present. Leaf Certificates shall have the *digitalSignature* bit set, and all other bits cleared. Non-Leaf Certificates shall have the *keyCertSign* bit set, may optionally have the *cRLSign* bit set, and shall have all other bits cleared.

3.1.3.4 Extended Key Usage (OID 2.5.29.37)

This extension shall be present and marked as critical. It shall contain the USB-IF issued OID 2.23.145.1.1 for the “USB-Auth” extended key usage, and may contain other OIDs.

3.1.3.5 Validity

The *notBefore* and *notAfter* fields indicate the time interval during which information regarding a Certificate’s validity is maintained. For Products, the validity times shall be ignored.

Certificate *notBefore* and *notAfter* validity times shall be specified using either ASN.1 GeneralizedTime for any year, or ASN.1 UTCTime for years prior to 2050.

It is recommended that the *notBefore* field be “19700101000000Z” (for 00:00 on 01-Jan-1970 UTC, which is POSIX epoch time). It is recommended that the *notAfter* field be “99991231235959Z” (for 23:59:59 on 31-Dec-9999 UTC, which is used for an unknown expiration time as defined in IETF-RFC-5280, Section 4.1.2.5).

3.1.3.6 USB-IF ACD (OID 2.23.145.1.2)

The USB-IF ACD extension is a custom Certificate extension for use with products compliant to this specification. It contains a binary object in the ACD format described in Appendix A.1. The binary object is encoded as an **ASN.1 DER** OCTET STRING with a maximum size of *MaxACDSize* bytes. The fields for a PD Product ACD are defined in Appendix A.2. The fields for a USB Product ACD are defined in Appendix A.3.

Leaf Certificates shall contain this extension. Non-Leaf Certificates shall not use this extension.

3.1.3.7 Additional Attributes and Extensions

Additional Certificate attributes and extensions defined in **X509v3** are allowed provided that the maximum Certificate size does not exceed *MaxLeafCertSize* for a Leaf Certificate or *MaxIntermediateCertSize* for a non-Leaf Certificate.

3.2 Certificate Chains

Certificates are grouped into Certificate Chains. A Certificate Chain is the binary (byte) concatenation of the fields shown in Table 3-1. A Certificate Chain shall not exceed *MaxCertChainSize* bytes.

Table 3-1: Certificate Chain Format

Offset	Field	Size	Description
0	<i>Length</i>	2	Total length of Certificate Chain in bytes including all fields in this table
2	<i>Reserved</i>	2	Set to zero
4	<i>RootHash</i>	32	32-byte SHA256 hash of the Root Certificate
36	<i>Certificates</i>	Length - 36	One or more ASN.1 DER -encoded X509v3 Certificates where the first Certificate is signed by the Root Certificate and each subsequent Certificate is signed by the preceding Certificate. The last Certificate is the Leaf Certificate.

Certificate Chains reside in positions called slots. Each slot shall either be empty or contain one complete certificate chain. A Product shall not contain more than 8 slots. Slots 0 through 3 shall only be used for Certificate Chains rooted with a USB-IF Root Certificate and shall not contain any other Certificate Chains. Slots 4 through 7 may be used for any additional Certificate Chains. The **ASN.1 DER** encoding of each individual certificate can be analyzed to determine its length.

3.2.1 Provisioning

Provisioning is the process by which a Product acquires one or more Certificate Chains. The procedure for provisioning a Product is outside the scope of this specification.

3.3 Private Keys

Each non-empty Certificate Chain slot in a Product corresponds to a private key whose corresponding public key is certified in the Leaf Certificate of that slot. The Product must have access to that private key. All private keys shall be unique and shall be generated and provisioned, and stored in a manner that adequately protects the confidentiality of the key. A private key used by one Product shall not be used by any other Products.

4 Authentication Protocol

There are three operations an Authentication Initiator can perform:

- Query an Authentication Responder for Certificate Chain digests
- Read a Certificate Chain from an Authentication Responder
- Challenge an Authentication Responder in order to verify its authenticity

An Authentication Initiator may initiate as many or as few of these operation as are needed to achieve the desired Authentication latency. In addition, an Authentication Initiator may initiate the operations in any order. For example, an Authentication Initiator that only uses slot 0 for Authentication, may first challenge a PD Product, and then initiate a Certificate Chain read if the target Certificate Chain is not already cached.

A Product shall not act as an Authentication Responder unless it contains a Certificate Chain in slot 0.

4.1 Digest Query

To query an Authentication Responder for Certificate Chain digests, an Authentication Initiator sends a [GET DIGESTS](#) Request as defined in Section 5.2.1. If an error condition is encountered, the Authentication Responder shall respond with the appropriate [ERROR](#) Response as defined in Section 4.4. Otherwise, the Authentication Responder shall respond with a [DIGESTS](#) Response as defined in Section 5.3.1. After receiving a [DIGESTS](#) Response, an Authentication Initiator can check to see if it has any of the Authentication Responder's Certificate Chains cached. This allows the Authentication Initiator to potentially skip reading a Certificate Chain and thus save time.

4.2 Certificate Chain Read

To read a Certificate Chain, or portion thereof, an Authentication Initiator sends a [GET CERTIFICATE](#) Request as defined in Section 5.2.2.

If an Authentication Responder receives a [GET CERTIFICATE](#) request that targets an offset that is outside the Certificate Chain (i.e. offset > length) or attempts to read beyond the length of the target Certificate Chain (i.e. (offset + length > Certificate Chain length)), then the Authentication Responder shall return an [ERROR](#) Authentication Response with *Param1* set to INVALID_REQUEST and *Param2* set to 00h.

If an error condition is encountered, the Authentication Responder shall respond with the appropriate [ERROR](#) Response as defined in Section 4.4. Otherwise, the Authentication Responder shall respond with a [CERTIFICATE](#) Response as described in Section 5.3.2.

4.3 Authentication Challenge

To challenge an Authentication Responder, an Authentication Initiator sends a [CHALLENGE](#) Request as defined in Section 5.2.3. If an error condition is encountered, the Authentication Responder shall respond with the appropriate [ERROR](#) Response as defined in Section 4.4. Otherwise, the Authentication Responder shall respond with a [CHALLENGE AUTH](#) Response as described in Section 5.3.3.

4.4 Errors and Alerts

4.4.1 Invalid Request

If an Authentication Responder receives an Authentication Request with one or more invalid fields, it shall respond to that Authentication Request with an [ERROR](#) Response that has *Param1* set to INVALID_REQUEST and *Param2* set to 00h.

4.4.2 Unsupported Protocol Version

If an Authentication Responder receives an Authentication Request that contains an unsupported Security Protocol Version in the *ProtocolVersion* field, it shall respond to that Authentication Request with an [ERROR](#) Response that has *ProtocolVersion* set to the minimum Security Protocol Version it supports, *Param1* set to UNSUPPORTED_PROTOCOL, and *Param2* set to the maximum Security Protocol Version it supports.

4.4.3 Busy

If an Authentication Responder receives an Authentication Request but is unable to meet either the timing requirements listed in Section 6.4 (for PD Products) or Section 7.4 (for USB Products), it shall respond to that Authentication Request with an [ERROR](#) Response that has *Param1* set to BUSY and *Param2* set to 00h.

4.4.4 Unspecified

If an Authentication Responder receives an Authentication Request then encounters an error that is not covered by the conditions above, it shall respond to that Authentication Request with an [ERROR](#) Response that has *Param1* set to UNSPECIFIED and *Param2* set to 00h.

5 Authentication Messages

Authentication Messages are used to convey information related to Authentication. An Authentication Message consists of a Message Header followed by a variable length (including zero) payload. The format for a Message Header is defined in Section 5.1.

There are two types of Authentication Messages: Authentication Requests and Authentication Responses. Authentication Requests are defined in Section 5.2. Authentication Responses are defined in Section 5.3.

Section 6 describes how Authentication Messages map onto PD messaging. Section 7 describes how Authentication Messages map onto USB transfers.

5.1 Header

All Authentication Messages shall start with the 4-byte header defined in Table 5-1.

Table 5-1: Authentication Message Header

Offset	Field	Size	Reference
0	<i>ProtocolVersion</i>	1	Section 5.1.1
1	<i>MessageType</i>	1	Section 5.1.2
2	<i>Param1</i>	1	Section 5.1.3
3	<i>Param2</i>	1	Section 5.1.4

5.1.1 USB Type-C Authentication Protocol Version

This field identifies which version of the USB Type-C Authentication Specification is being used. Table 5-2 shows the valid values for this field. A Product shall not use a USB Type-C Authentication Protocol Version value corresponding to a specification revision that it does not support.

Table 5-2: USB Type-C Authentication Protocol Version

Name	Value	Meaning
Reserved	00h	Reserved
V1.0	01h	USB Type-C Authentication Protocol Version 1.0
Reserved	02h-ffh	Reserved

It is intended in the future that Products support a contiguous range of USB Type-C Authentication Protocol Versions.

5.1.2 Message Type

This field identifies Authentication Message type and shall contain one of the Authentication Message Types listed in Table 5-3 or Table 5-9.

5.1.3 Param1

This field is used to pass a first 1-byte parameter. The contents of the parameter vary and are defined by Authentication Message type.

5.1.4 Param2

This field is used to pass a second 1-byte parameter. The contents of the parameter vary and are defined by Authentication Message type.

5.2 Authentication Requests

Authentication Requests are used to send a command to the recipient and/or retrieve data. Authentication Request types are listed in Table 5-3.

Table 5-3: Authentication Request Types

Value	Description
00h – 7Fh	Shall only be used for Authentication Responses
80h	Reserved
81h	GET_DIGESTS
82h	GET_CERTIFICATE
83h	CHALLENGE
84h - FFh	Reserved

5.2.1 GET_DIGESTS

This Request is used to retrieve Certificate Chain digests. The header for a GET_DIGESTS Request is defined in Table 5-4. A GET_DIGESTS Request has no payload.

Table 5-4: GET_DIGESTS Request Header

Offset	Field	Size	Value
0	<i>ProtocolVersion</i>	1	V1.0
1	<i>MessageType</i>	1	GET_DIGESTS
2	<i>Param1</i>	1	Reserved
3	<i>Param2</i>	1	Reserved

5.2.2 GET_CERTIFICATE

This Request is used to read a segment of a target Certificate Chain. The header for a GET_CERTIFICATE Request is defined in Table 5-5. The payload for a GET_CERTIFICATE Request is defined in Table 5-6.

Table 5-5: GET_CERTIFICATE Request Header

Offset	Field	Size	Value
0	<i>ProtocolVersion</i>	1	V1.0
1	<i>MessageType</i>	1	GET_CERTIFICATE
2	<i>Param1</i>	1	Slot number of the target Certificate Chain to read from. The value in this field shall be between 0 and 7 inclusive.
3	<i>Param2</i>	1	Reserved

Table 5-6: GET_CERTIFICATE Request Payload

Offset	Field	Size	Value
4	<i>Offset</i>	2	Offset in bytes from the start of the Certificate Chain to where the read request begins
6	<i>Length</i>	2	Length in bytes of the read request

5.2.3 CHALLENGE

This Request is used to initiate Authentication of a Product. The header for a CHALLENGE Request is defined in Table 5-7. The payload for a CHALLENGE Request is defined in Table 5-8.

Table 5-7: CHALLENGE Request Header

Offset	Field	Size	Value
0	<i>ProtocolVersion</i>	1	V1.0
1	<i>MessageType</i>	1	CHALLENGE
2	<i>Param1</i>	1	Slot number of the recipient's Certificate Chain that will be used for Authentication
3	<i>Param2</i>	1	Reserved

Table 5-8: CHALLENGE Request Payload

Offset	Field	Size	Description
4	<i>Nonce</i>	32	Random 32-byte nonce chosen by the Authentication Initiator

5.3 Authentication Responses

Authentication Responses are used to send a command and/or data to the recipient. Authentication Response types are listed in Table 5-9.

Table 5-9: Authentication Response Types

Value	Description
00h	Reserved
01h	DIGESTS
02h	CERTIFICATE
03h	CHALLENGE_AUTH
04h-7Eh	Reserved
7Fh	ERROR
80h – FFh	Shall only be used for Authentication Requests

5.3.1 DIGESTS

This Response is used by a Product to send Certificate Chain digests and report which slots contain valid Certificate Chain digests. The header for a DIGESTS Response is defined in Table 5-10. The Payload for a DIGESTS Response is defined in Table 5-11.

Table 5-10: DIGESTS Response Header

Offset	Field	Size	Value
0	<i>ProtocolVersion</i>	1	V1.0
1	<i>MessageType</i>	1	DIGESTS
2	<i>Param1</i>	1	Capabilities Field; set to 01h for this specification. All other values reserved.
3	<i>Param2</i>	1	Slot mask. The bit in position K of this byte is set if and only if slot number K contains a Certificate Chain for the protocol version in the <i>ProtocolVersion</i> field. (Bit 0 is the least significant bit of the byte.) The number of digests returned is equal to the number of bits set in this byte. The digests are returned in order of increasing slot number.

Table 5-11: DIGESTS Response Payload

Offset	Field	Size	Value
4	<i>Digest[0]</i>	32	32-byte SHA-256 digest of the first Certificate Chain
...
4 + (32 * (n - 1))	<i>Digest[n-1]</i>	32	32-byte SHA-256 digest of the last (n th) Certificate Chain

5.3.2 CERTIFICATE

This Response is used by a Product to send the requested segment of a Certificate Chain. The header for a CERTIFICATE Response is defined in Table 5-12. The payload for a CERTIFICATE Response is defined in Table 5-13.

Table 5-12: CERTIFICATE Response Header

Offset	Field	Size	Value
0	<i>ProtocolVersion</i>	1	V1.0
1	<i>MessageType</i>	1	CERTIFICATE
2	<i>Param1</i>	1	Slot number of the Certificate Chain returned
3	<i>Param2</i>	1	Reserved

Table 5-13: CERTIFICATE Response Payload

Offset	Field	Size	Value	Description
4	<i>CertChain</i>	<i>Length</i>	Data	Requested contents of target Certificate Chain

5.3.3 CHALLENGE_AUTH

This Response is used by a Product to respond to a CHALLENGE Request. The header for a CHALLENGE_AUTH Response is defined in Table 5-14. The payload for a CHALLENGE_AUTH Response is defined in Table 5-15.

Table 5-14: CHALLENGE_AUTH Response Header

Offset	Field	Size	Value
0	<i>ProtocolVersion</i>	1	V1.0
1	<i>MessageType</i>	1	CHALLENGE_AUTH
2	<i>Param1</i>	1	Slot number contained in the <i>Param1</i> field of the corresponding CHALLENGE Request
3	<i>Param2</i>	1	Slot mask. The bit in position K of this byte is set if and only if slot number K contains a Certificate Chain for the protocol version in the <i>ProtocolVersion</i> field. (Bit 0 is the least significant bit of the byte.)

Table 5-15: CHALLENGE_AUTH Response Payload

Offset	Field	Size	Value
4	<i>MinProtocolVersion</i>	1	Minimum protocol version supported by this Device
5	<i>MaxProtocolVersion</i>	1	Maximum protocol version supported by this Device
6	<i>Capabilities</i>	1	Set to 01h for this specification. All other values reserved
7	<i>Reserved</i>	1	Reserved
8	<i>CertChainHash</i>	32	32-byte SHA256 hash of the Certificate Chain used for Authentication
40	<i>Salt</i>	32	32-byte value chosen by the Authentication Responder <i>Note: the Salt can be random, fixed, or any other value</i>
72	<i>Context Hash</i>	32	See Section 6.5 for PD Products or Section 7.5 for USB Products
104	<i>Signature</i>	64	See Section 5.3.3.1

5.3.3.1 Signature

The *Signature* field in a CHALLENGE_AUTH Response contains a 64-byte **ECDSA** digital signature on the message contents listed in Table 5-16. The **ECDSA** signature is generated using values (r,s) with little-endian encoding, where r starts at offset 0 and s starts at offset 32. Each value is 32 Bytes with zero right-padding if necessary.

Table 5-16: Message Contents for ECDSA Digital Signature

Offset	Field	Size	Value
0	<i>ReqMsg</i>	36	Full contents (i.e. header and payload) of the corresponding CHALLENGE Request
36	<i>RespMsg</i>	104	Contents of the CHALLENGE_AUTH Response being signed excluding the <i>Signature</i> field

A message signer with a secure RNG can use non-deterministic **ECDSA**. A message signer without secure RNG capability can use deterministic **ECDSA**. In deterministic **ECDSA**, the random “k” value is derived from the hash of the message to be signed and a private key.

Note: in both deterministic and non-deterministic ECDSA, generating the “k” value has pitfalls and mistakes can lead to a leak of the private key. See RFC 6979 (available at: <https://tools.ietf.org/html/rfc6979>) for details.

5.3.4 ERROR

This Response is used by a Product to transmit error information. The header for an ERROR Response is defined in Table 5-17. An ERROR Response has no payload.

Table 5-17: ERROR Response Header

Offset	Field	Size	Description
0	<i>ProtocolVersion</i>	1	V1.0 ¹
1	<i>MessageType</i>	1	ERROR
2	<i>Param1</i>	1	Error Code. See Table 5-18.
3	<i>Param2</i>	1	Error Data. See Table 5-18.

Table 5-18: ERROR Codes

Error Code	Value	Description	Error Data
Reserved	00h	Reserved	Reserved
INVALID_REQUEST	01h	One or more Request fields are invalid	00h
UNSUPPORTED_PROTOCOL	02h	Requested Security Protocol Version is not supported	Maximum supported Security Protocol Version1
BUSY	03h	Device cannot respond now, but will be able to respond in the future	00h
UNSPECIFIED	04h	Unspecified error occurred	00h
Reserved	05h- EFh	Reserved	Reserved
Vendor Defined	F0h- FFh	Vendor defined	Vendor defined

¹Note: Minimum supported Security Protocol Version is returned in the *ProtocolVersion* field in the message header.

6 Authentication of PD Products

USBPD describes the mechanism for sending Authentication Messages over PD. The mapping of a security transfer to PD messages depends on whether or not the transfer exceeds *MaxExtendedMsgLen*. Note that *MaxExtendedMsgLen* is defined in **USBPD**.

6.1 Transfers less than or equal to *MaxExtendedMsgLen*

Security transfers where the Authentication Request and corresponding Authentication Response are each less than or equal to *MaxExtendedMsgLen* require only a single PD Message exchange. To initiate a security transfer, an Authentication Initiator sends a Security_Request PD Message carrying an Authentication Request in the PD Authentication Request Data Block (SRQDB). To complete a security transfer, an Authentication Responder sends a Security_Response PD Message carrying an Authentication Response in the PD Authentication Response Data Block (SRPDB).

For example, to retrieve Certificate Chain digests, an Authentication Initiator sends a Security_Request PD Message carrying a [GET DIGESTS](#) Authentication Message. The Authentication Responder then responds with a Security_Response PD Message carrying a [DIGESTS](#) Authentication Message (or appropriate [ERROR](#) Authentication Message).

6.2 Transfers greater than *MaxExtendedMsgLen*

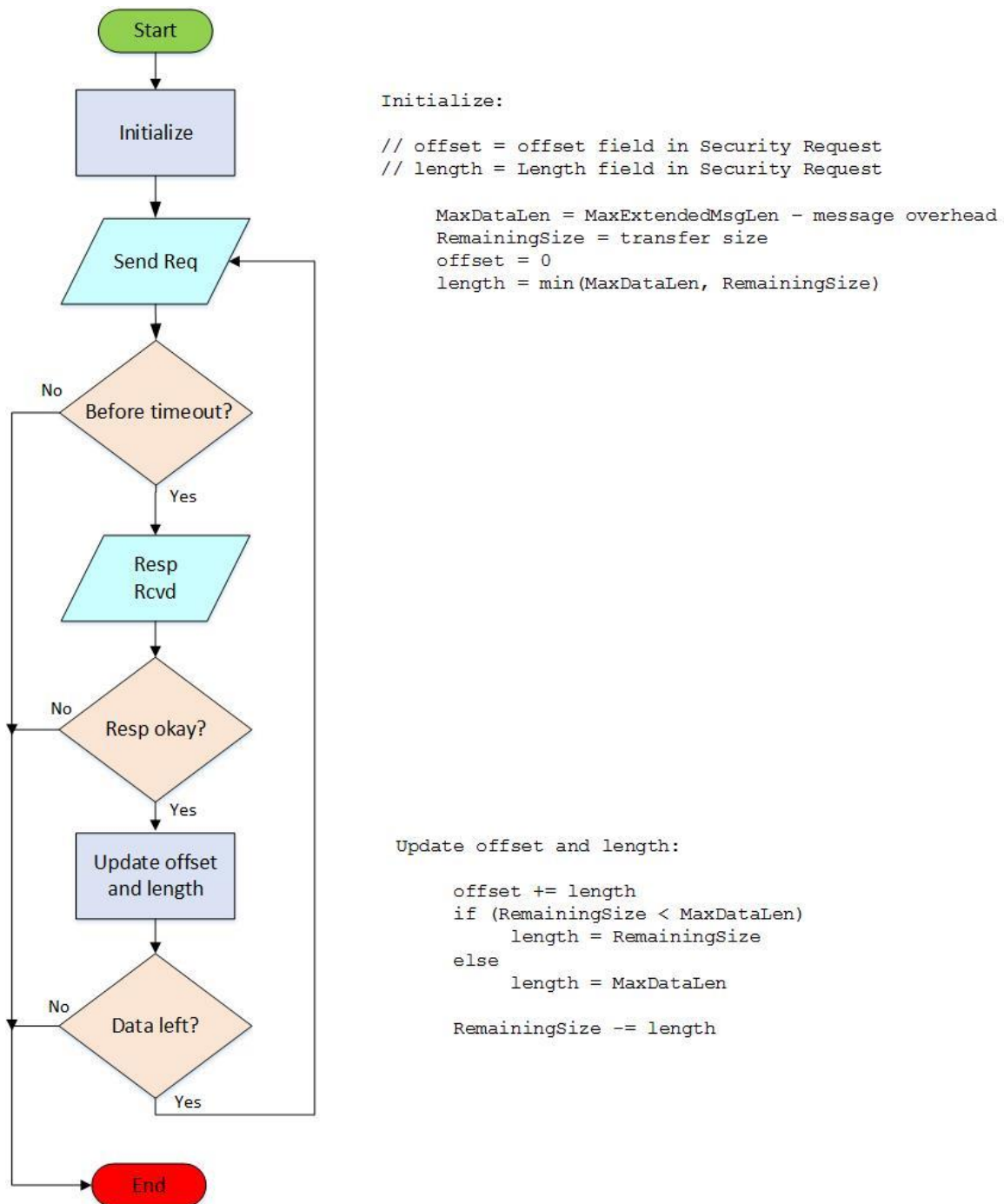
Security transfers where the Authentication Response is greater than *MaxExtendedMsgLen* require multiple PD message exchanges in order to fit into the PD Message framework.

An Authentication Initiator is responsible breaking up a security transfer into Authentication Messages that don't exceed *MaxExtendedMsgLen*. An Authentication Initiator does this by sending a series of Authentication Requests that target an incrementing segment of the security data to read in. Each Authentication Request is sent in the SRQDB of a Security_Request PD Message. Figure 6-1 shows the process by which an Authentication Initiator breaks up and completes a security transfer.

An Authentication Responder responds to an Authentication Request as shown in Figure 6-2. Each Authentication Response is sent as in the SRPDB of a Security_Response PD Message.

6.3 Examples

Figure 6-1 Example Security Transfer Process for an Authentication Initiator



Note: timeout values are listed in Table 6-1.

Figure 6-2 Example Security Transfer Process for an Authentication Responder

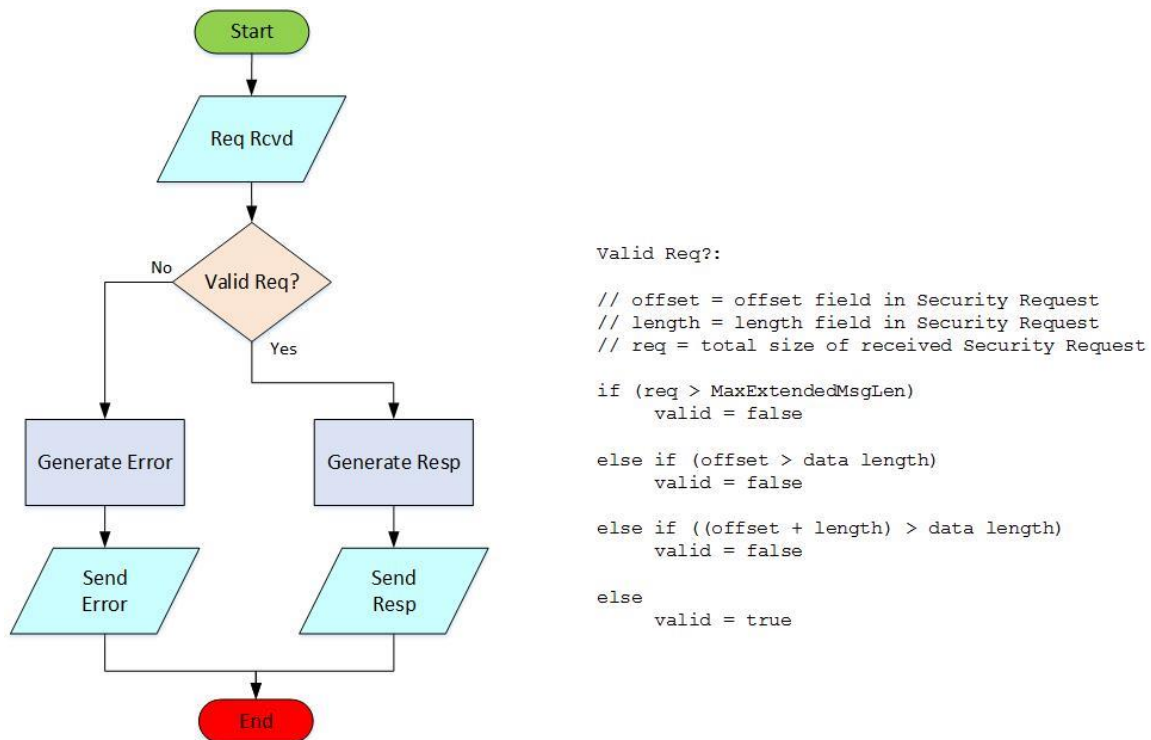
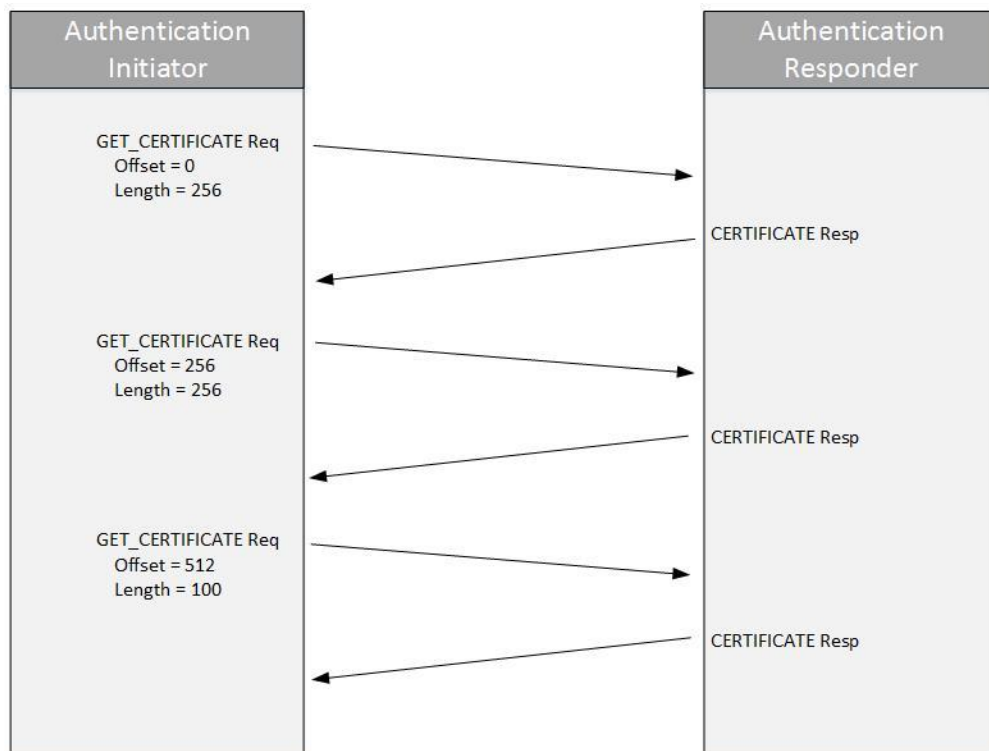


Figure 6-3 Example 612-Byte Certificate Chain Read



6.4 Timing Requirements for PD Security Extended Messages

6.4.1 Authentication Initiator

Table 6-1 shows the timing parameters that apply to an Authentication Initiator. Timeout values are measured from the moment the last bit of the Authentication Request is sent on the bus to the moment the last bit of the corresponding Authentication Response is received.

Table 6-1: Timeout Values for a PD Authentication Initiator

Parameter	Timeout Value		Description
	Unchunked	Chunked	
<i>tDigestRcvd</i>	40 ms	200 ms	Timeout for a GET DIGESTS Authentication Request.
<i>tCertRcvd</i>	40 ms	200 ms	Timeout for a GET CERTIFICATE Authentication Request.
<i>tChallengeAuthRcvd</i>	1000 ms	1200 ms	Timeout for a CHALLENGE Authentication Request.

If an Authentication Initiator does not receive an Authentication Response within *tDigestRcvd* of sending a [GET DIGESTS](#) Authentication Request, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

If an Authentication Initiator does not receive an Authentication Response within *tCertRcvd* of sending a [GET CERTIFICATE](#) Authentication Request to an Authentication Responder, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

If an Authentication Responder does not receive an Authentication Response within *tChallengeRespTimeout* of sending a [CHALLENGE](#) Authentication Request, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

6.4.2 Authentication Responder

Table 6-2 gives timing values that an Authentication Responder shall meet. Timing values are measured from the moment the last bit of an Authentication Request is received to the moment the last bit of the corresponding Authentication Response is sent on the bus.

Table 6-2: Timing Requirements for PD Authentication Responder

Parameter	Timing Value		Description
	Unchunked	Chunked	
<i>tDigestSent</i>	20 ms	135 ms	Maximum time between receiving a GET_DIGESTS Authentication Request and sending an Authentication Response.
<i>tCertSent</i>	20 ms	135 ms	Maximum time between receiving a GET_CERTIFICATE Authentication Request and sending an Authentication Response.
<i>tChallengeAuthSent</i>	520 ms	635 ms	Maximum time between receiving a CHALLENGE Authentication Request and sending an Authentication Response.

An Authentication Responder shall send an Authentication Response within *tDigestSent* of receiving a [GET_DIGESTS](#) Authentication Request.

An Authentication Responder shall send an Authentication Response within *tCertSent* of receiving a [GET_CERTIFICATE](#) Authentication Request.

An Authentication Responder shall send an Authentication Response within *tChallengeAuthSent* of receiving a [CHALLENGE](#) Authentication Request.

6.5 Context Hash

The *Context Hash* field in a [CHALLENGE_AUTH](#) Authentication Response shall be zero for PD Sources, Sinks and Cable Plugs. This field shall also be zero for PD Alternate Mode devices.

7 Authentication of USB Products

A USB Host can use the architecture and methods defined in this specification to authenticate a USB Device. A USB Host that authenticates a USB Device takes the role of Authentication Initiator with the USB Device taking the role of Authentication Responder. A USB Device shall not act as an Authentication Initiator.

This section describes the additional framework needed for USB Device Authentication.

7.1 Descriptors

The descriptors in this section are used to describe the Authentication capabilities of a USB Device.

7.1.1 Authentication Capability Descriptor

This descriptor is used to advertise the Authentication capabilities and features a USB Device supports. This descriptor shall be returned as part of the BOS Descriptor set for a USB Device that supports Authentication.

Table 7-1: Authentication Capability Descriptor

Offset	Field	Size	Value	Description								
0	<i>bLength</i>	1	Number	Size of this Descriptor in Bytes								
1	<i>bDescriptorType</i>	1	Constant	Descriptor type: DEVICE CAPABILITY (16)								
2	<i>bDevCapabilityType</i>	1	Number	AUTHENTICATION								
3	<i>bmAttributes</i>	1	Bitmap	Bitmap encoding of supported features. <table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Set to 1 to indicate that firmware can be updated. Otherwise, set to zero.</td></tr><tr><td>1</td><td>Set to 1 to indicate that Device changes interfaces when updated. Otherwise, set to zero.</td></tr><tr><td>7:2</td><td>Reserved. Shall be set to zero.</td></tr></table>	Bit	Description	0	Set to 1 to indicate that firmware can be updated. Otherwise, set to zero.	1	Set to 1 to indicate that Device changes interfaces when updated. Otherwise, set to zero.	7:2	Reserved. Shall be set to zero.
Bit	Description											
0	Set to 1 to indicate that firmware can be updated. Otherwise, set to zero.											
1	Set to 1 to indicate that Device changes interfaces when updated. Otherwise, set to zero.											
7:2	Reserved. Shall be set to zero.											
4	<i>bcdProtocolVersion</i>	1	Number	USB Type-C Authentication Protocol Version								
5	<i>bcdCapability</i>	1	Number	Value is the same as <i>Param1</i> in a DIGESTS Authentication Response								

Table 7-2: Authentication Capability Descriptor Types

Authentication Capability Descriptor Type	Value
AUTHENTICATION	0Eh

7.2 Mapping Authentication Messages to USB

Authentication Messages are transmitted over USB using the control requests defined in this section. Two additional USB Standard Request Codes used for USB Authentication are defined in Table 7-3.

Table 7-3: Authentication Message *bRequest* Values

Authentication Message <i>bRequest</i>	Value
AUTH_IN	18h
AUTH_OUT	19h

7.2.1 Authentication IN

This control request is used by a USB Host to transfer an Authentication Message without payload to a USB Device. This request initiates a control IN transfer.

The *wValue* and *wIndex* fields of an Authentication IN control packet contain the 4-byte Authentication Message Header described in Section 5.1. Table 7-5 shows the mapping of an Authentication Message Header onto the *wValue* and *wIndex* fields.

Table 7-4: Authentication IN Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000b	AUTH_IN	Authentication Message Header		Varies	Varies

Table 7-5: Authentication Message Header Mapping

Offset	Field	Size	Value	Description
0	<i>ProtocolVersion</i>	1	Number	Maps to high byte of <i>wValue</i>
1	<i>MessageType</i>	1	Number	Maps to low byte of <i>wValue</i>
2	<i>Param1</i>	1	Number	Maps to high byte of <i>wIndex</i>
3	<i>Param2</i>	1	Number	Maps to low byte of <i>wIndex</i>

The behavior of a USB Device is not specified if *wLength*, *wIndex*, or *wValue* for a particular Response type do not match the values set forth in this section. Otherwise, the following behavior is expected for the given USB Device state:

Default	Not specified.
Address	Request is valid and Device shall respond.
Configured	Request is invalid and Device shall respond with a Request Error.

7.2.2 Authentication OUT

This control request is used by a USB Host to transfer an Authentication Message with payload to a USB Device. This request initiates a control OUT transfer.

Table 7-6: Authentication OUT Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000b	AUTH_OUT	Authentication Message Header		Varies	Varies

The *wValue* and *wIndex* fields of an Authentication OUT control packet contain the 4-byte Authentication Message Header described in Section 5.1. Table 7-5 shows the mapping of an Authentication Message Header onto the *wValue* and *wIndex* fields.

The behavior of a USB Device is not specified if *wLength*, *wIndex*, or *wValue* for a particular Authentication Request type do not match the values set forth in this section. Otherwise, the following behavior is expected for the given USB Device state:

Default	Not specified.
Address	Request is valid and Device shall respond.
Configured	Request is invalid and Device shall respond with a Request Error.

7.3 Authentication Protocol

A USB Host can perform the Authentication operations described in Section 4. This section describes how each of the operations are performed.

7.3.1 Digest Query

A USB Host uses the [GET DIGESTS](#) Request defined in Section 5.2.1 to retrieve Certificate Chain digests from a USB Device. To send a [GET DIGESTS](#) Request, a USB Host initiates an AUTH_IN Control Transfer with the values shown in Table 7-7.

Table 7-7: GET_DIGESTS Authentication IN Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000b	AUTH_IN	See Table 5-4 for values and Table 7-5 for mapping.		260	DIGEST Response

In response to a [GET DIGESTS](#) Request, a USB Device returns either a DIGESTS Response as described in 5.3.1 or [ERROR](#) Response as described in Section 5.3.4.

7.3.2 Certificate Read

A USB Host uses an AUTH_OUT control transfer followed by an AUTH_IN control transfer to read a Certificate Chain or part thereof from a USB Device. The AUTH_OUT control transfer contains a [GET CERTIFICATE](#) Request as shown in Table 7-8.

Table 7-8: GET_CERTIFICATE Authentication OUT Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000b	AUTH_OUT	See Table 5-5 for values and Table 7-5 for mapping.		4	See Table 5-6

After successfully completing the AUTH_OUT control transfer, the USB Host initiates an AUTH_IN Control transfer to retrieve the [CERTIFICATE](#) Response from the USB Device. The AUTH_IN Control transfer contains the values in Table 7-9 where the high byte of *wValue* contains the *ProtocolVersion* field, the low byte of *wValue* contains the *MessageType* field in Table 5-12, and the *wLength* field contains the *length* value in the [GET CERTIFICATE](#) Response contained in the previous AUTH_OUT Control transfer plus 4 (to account for the Authentication Message Header).

Table 7-9: CERTIFICATE Authentication IN Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000b	AUTH_IN	See Table 5-12	0	length + 4	CERTIFICATE Response

In response to the AUTH_IN Control transfer above, a USB Device returns either a [CERTIFICATE](#) Response as described in Section 5.3.2 or [ERROR](#) Response as described in Section 5.3.4.

7.3.3 Authentication Challenge

A USB Host uses an AUTH_OUT control transfer followed by an AUTH_IN control transfer to authenticate a USB Device. The AUTH_OUT control transfer contains a [CHALLENGE](#) Request as shown in Table 7-10.

Table 7-10: CHALLENGE Authentication OUT Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000b	AUTH_OUT	See Table 5-7 for values and Table 7-5 for mapping.		32	See Table 5-8

After successfully completing the AUTH_OUT control transfer, the USB Host initiates an AUTH_IN control transfer to retrieve the [CHALLENGE AUTH](#) Response from the USB Device. The AUTH_IN control transfer contains the values in Table 7-11 where the high byte of *wValue* contains the *ProtocolVersion* field and the low byte of *wValue* contains the *MessageType* field in Table 5-14.

Table 7-11: CHALLENGE_AUTH Authentication IN Control Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000b	AUTH_IN	See Table 5-14	0	132	CHALLENGE AUTH Response

In response to the AUTH_IN control transfer above, a USB Device returns either a [CHALLENGE AUTH](#) Response as described in Section 5.3.3 or [ERROR](#) Response as described in Section 5.3.4.

7.3.4 Errors

If a USB Device encounters an Authentication-related error condition during an AUTH_IN control transfer, it shall respond with an [ERROR](#) Response. If a USB Device encounters an Authentication-related error condition during an AUTH_OUT control transfer, it shall respond to the next AUTH_IN control transfer with an [ERROR](#) Response. The format for an [ERROR](#) Response is defined in Section 5.3.4.

7.4 Timing Requirements for USB

All Authentication Message exchanges over USB shall follow the timing for control transfers set forth in **USB2.0** and **USB3.1**.

In addition, a USB Host shall use the timeout values in Table 7-12 .

Table 7-12: Authentication Initiator Timeout Values

Parameter	Timeout Value	Description
<i>tDigestIN</i>	100 ms	Timeout for a GET DIGESTS Authentication Request.
<i>tCertOUT</i>	100 ms	Timeout for an AUTH_OUT control transfer carrying a GET CERTIFICATE Authentication Request.
<i>tCertIN</i>	500 ms	Timeout for an AUTH_IN control transfer carrying a CERTIFICATE Authentication Response.
<i>tChallengeOUT</i>	100 ms	Timeout for an AUTH_OUT control transfer carrying a CHALLENGE Authentication Request.
<i>tChallengeIN</i>	600 ms	Timeout for an AUTH_IN control transfer carrying a CHALLENGE AUTH Authentication Response.

If a USB Host does not receive an Authentication Response within *tDigestIN* of sending a [GET DIGESTS](#) Authentication Request, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

If a USB Host does not receive an ACK within *tCertOUT* of sending a [GET CERTIFICATE](#) Authentication Request to an Authentication Responder, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

If a USB Host does not receive an Authentication Response within *tCertIN* of sending a [CERTIFICATE](#) Authentication Request to an Authentication Responder, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

If a USB Host does not receive an ACK within *tChallengeOUT* of sending a [CHALLENGE](#) Authentication Request to an Authentication Responder, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

If a USB Host does not receive an Authentication Response within *tChallengeIN* of sending a [CHALLENGE AUTH](#) Authentication Request to an Authentication Responder, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

7.5 Context Hash

This field contains a 32-byte **SHA256** hash of the following USB Descriptor data (as defined in **USB2.0** and **USB3.1**) for current operating speed, concatenated together in following order:

1. Device Descriptor
2. Complete BOS Descriptor (if present)
3. Complete Configuration 1 Descriptor
4. Complete Configuration 2 Descriptor (if present)
5. ...
6. Complete Configuration n Descriptor (if present)

The contents of each descriptor listed above shall match that which the device presents during enumeration at the USB Device's current connection.

8 Protocol Constants

Table 8-1: Protocol Constants

Constant	Value
<i>MaxLeafCertSize</i>	640 bytes
<i>MaxIntermediateCertSize</i>	512 bytes
<i>MaxACDSize</i>	128 bytes
<i>MaxCertChainSize</i>	4096 bytes

A ACD

A.1. ACD Formatting

ACD formatting consists of a sequence of zero or more TLV fields that start at the first byte of a binary object. The general format for a TLV field is set out in Table A-1. TLV types are listed in Table A-2 and defined in more detail below.

Each TLV type shall occur no more than once. Each TLV shall appear in increasing order by TLV value.

Table A-1: TLV General Format

Offset	Field	Size	Description
0	Type	1	TLV Type
1	Length	1	Number of bytes in Data field
2	Data	Length	Determined by TLV type

Table A-2: TLV Types

Value	Name
00h	VERSION
01h	XID
02h	POWER_SOURCE_CAPABILITIES
03h	POWER_SOURCE_CERTIFICATIONS
04h	CABLE_CAPABILITIES
05h	SECURITY_DESCRIPTION
06h - FCh	Reserved
FDh	PLAYPEN
FEh	VENDOR_EXTENSION
FFh	EXTENSION

A.1.1. Version TLV

This TLV is used to specify the ACD type and version. Valid ACD Versions are listed in Table A-4.

Table A-3: Version TLV Fields

Offset	Field	Size	Value
0	Type	1	VERSION
1	Length	1	2
2	Data	2	See Figure A-1

Figure A-1: Bitmap of Version TLV Data

Bit[7]	Bit[6]	Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
ACD Version							
Bit[15]	Bit[14]	Bit[13]	Bit[12]	Bit[11]	Bit[10]	Bit[9]	Bit[8]
USB*	PD*	Cable*	Reserved				

* Bit 15 is set to 1 for a USB Product ACD. Bit 14 is set to 1 for a PD Product ACD. Bit 13 is set to 1 for a USB Type-C Cable.

Table A-4: ACD Version Encoding

Value	Description
0	Adheres to the ACD format defined in USB Type-C Authentication Specification Revision 1.0
1 - 255	Reserved

A.1.2. XID TLV

This TLV is used to convey the 32-bit XID originating from the USB-IF for the purposes of compliance testing (http://www.usb.org/developers/compliance/request_XID/).

Table A-5: XID TLV Fields

Offset	Field	Size	Value
0	Type	1	XID
1	Length	1	4
2	Data	4	Vendor-selected from a block of 32-bit values assigned by the USB-IF

A.1.3. Power Source Capabilities TLV

This TLV is used to specify the power source capabilities of a PD power source.

Table A-6: Power Source Capabilities TLV Fields

Offset	Field	Size	Value
0	Type	1	POWER_SOURCE_CAPABILITIES
1	Length	1	Varies
2	Data	Length	See Table A-7

Table A-7: Power Source Capabilities TLV Data

Offset	Field	Size	Description
0	Version	1	Upper nibble contains the TLV Version Lower nibble contains the PD Revision. See USBPD Section 6.2.1.1.5 (Specification Revision).
1	FW Version	1	See USBPD , Section 6.5.1.2 (Product ID Field) and Table 6-37 (Source Capabilities Extended Data Block).
2	HW Version	1	See USBPD , Section 6.5.1.3 (Hardware Version Field).
3	Voltage Regulation	1	See USBPD , Section 6.5.1.4 (Voltage Regulation Field).
4	Hold Up time	1	See USBPD , Section 6.5.1.5 (Holdup Time Field).
5	Compliance	1	See USBPD , Section 6.5.1.6 (Compliance Field).
6	Touch Current	1	See USBPD , Section 6.5.1.7 (Touch Current).
7	Reserved	1	Reserved.
8	Peak Current 1	2	See USBPD , Section 6.5.1.8 (Peak Current).
10	Peak Current 2	2	See USBPD , Section 6.5.1.8 (Peak Current).
12	Peak Current 3	2	See USBPD , Section 6.5.1.8 (Peak Current).
14	Touch Temp	1	See USBPD , Section 6.5.1.9 (Touch Temp).
15	Source Inputs	1	See USBPD , Section 6.5.1.10 (Source Inputs).
16	Batteries	1	See USBPD , Section 6.5.1.11 (Batteries).
17	Num PDOs	1	See USBPD , Section 6.2.1.1.2 (Number of Data Objects).
18	PDOs	Num PDOs * 4	See USBPD , Section 6.4.1.2 (Source Capabilities Message).

A.1.4. Power Source Certifications TLV

This TLV contains a bitmap attesting to the certifications a PD Product has achieved.

Table A-8: Power Source Certifications TLV Fields

Offset	Field	Size	Value
0	Type	1	POWER_SOURCE_CERTIFICATIONS
1	Length	1	Varies
2	Data	Length	Reserved

A.1.5. Cable Capabilities TLV

Table A-9: Cable Capabilities TLV Fields

Offset	Field	Size	Value
0	Type	1	POWER_SOURCE_CERTIFICATIONS
1	Length	1	6
2	Data	6	See Table A-10

Table A-10: Cable Capabilities TLV Data

Offset	Field	Size	Description
0	Version	1	Upper nibble contains the TLV Version Lower nibble contains the PD Revision. See USBPD , Section 6.2.1.1.5 (Specification Revision).
1	Product Type	1	See USBPD , Section 6.4.4.3.1.1.4 (Product Type Cable Plug).
2	Cable VDO	4	See USBPD , Table 6-30 (Passive Cable VDO) and Table 6-31 (Active Cable VDO). See also USBPD Section 6.4.4.3.1.2 (Cable VDO).

A.1.6. Security Description TLV

This TLV contains information about the secure components of a Product and their trustworthiness. The certifications claimed in this field shall be relevant and appropriate to the security functions used in the product.

Table A-11: Security Description TLV Fields

Offset	Field	Size	Value
0	Type	1	SECURITY_DESCRIPTION
1	Length	1	6
2	Data	6	See Table A-12

Table A-12: Security Data

Offset	Field	Size	Value
0	FIPS/ISO Identifier	1	See Section A.1.6.1
1	Common Criteria Identifier	2	See Section A.1.6.2
3	Security Analysis Identifier	1	See Section A.1.6.3
4	IC Vendor Identifier	2	See Section A.1.6.4

A.1.6.1 FIPS/ISO Identifier

This field encodes the NIST-FIPS-140-2 or ISO-19790 security level of the Product. A Product uses the values listed in Table A-13 to indicate which certification it has received (if any).

Table A-13: FIPS/ISO Level Identifiers

Value	Description
0	No FIPS/ISO certification
1	ISO-19790 (ed1: 2006), NIST-FIPS 140-2, Level 1
2	ISO-19790 (ed1: 2006), NIST-FIPS 140-2, Level 2
3	ISO-19790 (ed1: 2006), NIST-FIPS 140-2, Level 3
4	ISO-19790 (ed1: 2006), NIST-FIPS 140-2, Level 4
5-8	Reserved
9	ISO-19790 (ed2: 2012), Level 1
10	ISO-19790 (ed2: 2012), Level 2
11	ISO-19790 (ed2: 2012), Level 3
12	ISO-19790 (ed2: 2012), Level 4
13 - 255	Reserved

A.1.6.2 Common Criteria Identifier

This field encodes *Common Criteria* information for a product in the format shown in Figure A-1.

Figure A-2: Bitmap of the Common Criteria Identifier

Bit[7]	Bit[6]	Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
Protection Profile Encoding		EAL Level			Vulnerability Assessment		
Bit[15]	Bit[14]	Bit[13]	Bit[12]	Bit[11]	Bit[10]	Bit[9]	Bit[8]
Development Security		CM*	Certification Year				

* CM: Certificate Maintenance

A.1.6.2.1 Vulnerability Assessment

The vulnerability assessment (AVA_VAN) certifies a resistance to attackers with certain attack potential (AVA_VAN Definition: “Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components (September 2012, Version 3.1, Revision 4)”, <https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf>, see “8 Evaluation assurance levels” and “16 Class AVA: Vulnerability assessment”). USB Products should achieve AVA_VAN.5 to provide a sufficient level of protection. Vulnerability assessment encodings are shown in Table A-14.

Table A-14: Vulnerability Assessment

Value	Description
0	No vulnerability assessment performed
1	AVA_VAN.1 Vulnerability survey
2	AVA_VAN.2 Vulnerability analysis
3	AVA_VAN.3 Focused vulnerability analysis
4	AVA_VAN.4 Methodical vulnerability analysis
5	AVA_VAN.5 Advanced methodical vulnerability analysis

A.1.6.2.2 EAL Level

The EAL level defines the Evaluation Assurance Level (EAL) of the USB Product and indicates the confidence and amount of testing performed by the certification agency. EAL levels encodings are shown in Table A-15.

Table A-15: EAL Encodings

Value	Description
0	No Common Criteria certification performed
1	EAL1: Functionally Tested
2	EAL2: Structurally Tested
3	EAL3: Methodically Tested and Checked
4	EAL4: Methodically Designed, Tested and Reviewed
5	EAL5: Semi-formally Designed and Tested
6	EAL6: Semi-formally Verified Design and Tested
7	EAL7: Formally Verified Design and Tested

A.1.6.2.3 Protection Profile

An appropriate protection profile for **Common Criteria** evaluation should be used. The list of appropriate protection profiles is available on the Common Criteria Portal (available at: <https://www.commoncriteriaportal.org/pps/>). The two protection profiles for “ICs, Smart Cards and Smart-card related devices and systems” are: BSI-CC-PP-0084-2014 and BSI-PP-0035-2007. The different encodings for the Protection Profile field are provided in Table A-16.

Table A-16: Protection Profile Encoding

Value	Description
0	No Common Criteria evaluation performed
1	Security IC platform protection profile BSI-PP-0035-2007
2	Security IC platform protection profile BSI-CC-PP-0084-2014
3	Other protection profile listed in Common Criteria

A.1.6.2.4 Development Security

The life cycle support development security (ALC_DVS) is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE (Target of Evaluation). It includes the physical security of the development location(s) and controls on the selection and hiring of development staff. (ALC_DVS Definition: “Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components (September 2012, Version 3.1, Revision 4)”, <https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf>, see “14.4 Development Security”).

Table A-17: Development Security

Value	Description
0	No ALC_DVS compliant development security performed
1	ALC_DVS.1 Identification of Security Measures
2	ALC_DVS.2 Sufficiency of Security Measures
3	Reserved

A.1.6.2.5 Certification Maintenance

Declares that, at the time of the corresponding Product's production completion date, the specified **Common Criteria** certification with augmentations was still valid and in force, from either the original certification or extended through the use of certification maintenance packages accepted by **Common Criteria** for the original certification.

Table A-18: Certification Maintenance

Value	Description
0	No Declaration of Common Criteria certification validity at production completion date
1	Common Criteria certification valid at Product's production completion date

A.1.6.2.6 Certification Year

Specifies the year that the corresponding **Common Criteria** certificate was obtained. The year value is specified as (year – 2010) and encoded into 5 bits. This allows for certification years in the range from 2010 to 2041.

A.1.6.3 Security Analysis Identifier

This field expresses the degree of attack resistance that was established outside of the FIPS or **Common Criteria** regime either by internal testing carried out by the vendor (or its supplier), testing carried out by an external lab, or by deriving the degree of assurance from an alternative or future certification. The level of attack resistance is measured according to the rating in the JIL/JHAS document “Application of Attack Potential to Smartcards” (JIL/JHAS: “Application of Attack Potential to Smartcards - Joint Interpretation Library (Version 2.9, January 2013)”, <http://www.sogisportal.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v2-9.pdf>). The encoding is described in Figure A-2.

Figure A-3: Bitmap of the Security Analysis Identifier

Bit[7]	Bit[6]	Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
Reserved	JHAS/JIL resistance			Testing method			

A.1.6.3.1 Testing Method

A vendor can use the Bit fields shown in Table A-19 to express how the stated level of attack resistance was determined. Possible options are internal testing, testing by an external lab,

or the derivation of the attack resistance from a certification. This can either be **Common Criteria**, or FIPS, but also different or future security certification and evaluation standards. It is possible to set multiple bits, e.g., if internal testing, external testing, and certification was performed. If no bits are set, then no testing or security evaluation is claimed.

Table A-19: Testing Method Encoding

Bit	Description
Bit[0]	Internally tested by vendor
Bit[1]	Tested by external lab
Bit[2]	Derived from certification (not exclusive to Common Criteria or FIPS)
Bit[3]	Reserved

A.1.6.3.2 JIL/JHAS Resistance

The encodings for the resistance against attackers with a certain potential are shown in Table A-20. The rating is based on the JIL/JHAS document “Application of Attack Potential to Smartcards” (JIL/JHAS: “Application of Attack Potential to Smartcards - Joint Interpretation Library (Version 2.9, January 2013)”, <http://www.sogisportal.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v2-9.pdf>) and in case this document was not used for evaluation or internal testing it is up to the vendor to rate his product appropriately.

Table A-20: Vulnerability Assessment

Value	Description
0	No vulnerability assessment performed
1	No rating
2	Basic
3	Enhanced-Basic
4	Moderate
5	High
6 -7	Reserved

A.1.6.4 IC Vendor Identifier

This 2-byte field shall contain either the USB-IF-assigned VID that identifies the IC vendor or zero if not used.

A.1.7. Playpen TLV

This TLV is used for development purposes only. It shall not be used or interpreted by any Products.

Table A-21: Playpen TLV Fields

Offset	Field	Size	Value
0	Type	1	PLAYPEN
1	Length	1	To be set by developer
2	Data	Varies	To be set by developer

A.1.8. Vendor Extension TLV

This TLV contains a vendor proprietary information. The first two bytes of the Data field shall always contain the Vendor ID of the vendor defining the field.

Table A-22: Vendor Extension TLV Fields

Offset	Field	Size	Value
0	Type	1	VENDOR_EXTENSION
1	Length	1	Vendor defined
2	Data	Varies	Vendor defined

A.1.9. Extension TLV

This TLV is used to address future exhaustion of the TLV space. There are currently no Extension types and this TLV type shall not be used.

Table A-23: Extension TLV Fields

Offset	Field	Size	Value
0	Type	1	EXTENSION
1	Length	1	Length of Data field
2	Data	Varies	Extension type data

A.2. ACD for a PD Product

The ACD for a PD Product is defined in Table A-24. All TLV Types marked as “Required” shall be present. Types marked “N/A” are not allowed and shall not be used.

Table A-24: PD Product ACD TLVs

Value	Name	PD Source/Sink	USB Type-C Cable
00h	VERSION	Required	Required
01h	XID	Required	Required
02h	POWER_SOURCE_CAPABILITIES	Required	N/A
03h	POWER_SOURCE_CERTIFICATIONS	Reserved	N/A
04h	CABLE_CAPABILITIES	N/A	Required
05h	SECURITY_DESCRIPTION	Required	Required
06h - FCh	Reserved	--	--
FDh	PLAYPEN	Optional	Optional
FEh	VENDOR_EXTENSION	Optional	Optional
FFh	EXTENSION	Optional	Optional

A.3. ACD for a USB Product

The ACD for a USB Product is defined in Table A-25. All TLV Types marked as “Required” shall be present. Types marked “N/A” are not allowed and shall not be used. Types marked as “Conditional” may only be used in PDUSB Products.

Table A-25: USB Product ACD TLVs

Value	Name	USB
00h	VERSION	Required
01h	XID	Optional
02h	POWER_SOURCE_CAPABILITIES	Conditional
03h	POWER_SOURCE_CERTIFICATIONS	Conditional
04h	CABLE_CAPABILITIES	N/A
05h	SECURITY_DESCRIPTION	Required
06h - FCh	Reserved	--
FDh	PLAYPEN	Optional
FEh	VENDOR_EXTENSION	Optional
FFh	EXTENSION	Optional

B Sample Certificate Chain

The following certificate chain is included for illustrative purposes only.

B.1. Root Certificate

```
-----BEGIN CERTIFICATE-----
MIIBbzCCARagAwIBAgIBATAKBggqhkJOPQQDAjAkMRIwEAYDVQQKDA1VU0ItSUYg
Q0ExDjAMBgNVBAMMBVVTQjo6MCIYDzE5NzAwMTAxMDAwMDAwWhgPOTk5OTEyMzEy
MzU5NTlaMCQxEjAQBgNVBAoMCVVTQilJRiBDQTEOMAwGA1UEAwwFVVNCOjowWTAT
BgcqhkJOPQIBBggqhkJOPQMBBwNCAAQ7lPbuOmlousDtizZuCTgoXJxVbGmMOBOL
+zzQ/FfCrvk+6/FPFDfhcfWNcCO10oP6SKKUwML5MrfKbGs9TbT+ozUwMzAPBgNV
HRMBAf8EBTADAQH/MASGA1UdDwQEAwIBBjATBgNVHSUBAf8ECTAHBgVngREBATAK
BggqhkJOPQQDAGNHADBEAiA6efKzT06o4701I03AqI5B6gMtFGJg4Kc4iLNVPxB1
7QIgPKcUTGPPZFt9mmpX/Gva/RlRSZ2TWq9Jh3iBlpFAY8=
-----END CERTIFICATE-----
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
  Signature Algorithm: ecdsa-with-SHA256
    Issuer: O=USB-IF CA, CN=USB::
    Validity
      Not Before: Jan  1 00:00:00 1970 GMT
      Not After : Dec 31 23:59:59 9999 GMT
    Subject: O=USB-IF CA, CN=USB::
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:3b:94:f6:ee:3a:69:68:ba:c0:ed:89:96:6e:09:
        38:28:5c:9c:55:6c:69:8c:38:13:8b:fb:3c:d0:fc:
        57:c2:ae:f9:3e:eb:f1:4f:14:37:e1:71:f5:8d:70:
        23:b5:d2:83:fa:48:a2:94:c0:c2:f9:32:b7:ca:6c:
        6b:3d:4d:b4:fe
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Key Usage:
        Certificate Sign, CRL Sign
      X509v3 Extended Key Usage: critical
        2.23.145.1.1
  Signature Algorithm: ecdsa-with-SHA256
    30:44:02:20:3a:79:f2:b3:4f:4e:a8:e3:bd:25:23:4d:c0:a8:
    8e:41:ea:03:2d:14:62:60:e0:a7:38:88:b3:55:3f:10:65:ed:
    02:20:3c:a7:14:4c:63:cf:64:5b:7d:9a:6a:57:fc:6b:da:fd:
    19:51:49:9d:93:5a:af:49:26:1d:e2:06:5a:45:01:8f
```

B.2. First Intermediate Certificate

-----BEGIN CERTIFICATE-----

```
MIIBeTCCAR6gAwIBAgICAgEwCgYIKoZIZj0EAwIwJDESMBAGA1UECgwJVVNCLU1G
IENBMQ4wDAYDVQQDDAVVU0I6OjAiGA8xOTcwMDEwMTAwMDAwMFoYDzk5OTkxMjMx
MjM1OTU5WjArMRUwEwYDVQQKDAxPcmdOYW1lIEluYy4xEjAQBgNVBAMMCVVTQjox
YTBhOjBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABGXClXRpKfmRyyDlTf63k1os
GM3DGWeii2ESRzCursQ5TiwDm3vBZrhlmtQxLy2vtI1/Q7m60fbSJw0hjiQ7fG6j
NTAzMA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQDAgEGMBMGAlUdJQEB/wQJMAcG
BWeBEQEBAoGCCqGSM49BAMCA0kAMEYCIQCXX2fyWYjLftFuZDj1OAhCGrzK+SVI
jnIsybIDUzaElAIhAOHomRtJ6icSSdQl99S89pS+fvO3+6Op3pIdD+QNdf6I
-----END CERTIFICATE-----
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 513 (0x201)

Signature Algorithm: ecdsa-with-SHA256

Issuer: O=USB-IF CA, CN=USB::

Validity

Not Before: Jan 1 00:00:00 1970 GMT

Not After : Dec 31 23:59:59 9999 GMT

Subject: O=OrgName Inc., CN=USB:1a0a:

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:65:c2:95:74:69:29:f9:91:cb:20:e5:4d:fe:b7:

93:5a:2c:18:cd:c3:19:67:a2:8b:61:12:47:30:94:

ae:c4:39:4e:2c:03:9b:7b:c1:66:b8:65:99:3a:b1:

2f:2d:af:b4:8d:7f:43:b9:ba:d1:f6:d2:27:0d:21:

8e:24:3b:7c:6e

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Key Usage:

Certificate Sign, CRL Sign

X509v3 Extended Key Usage: critical

2.23.145.1.1

Signature Algorithm: ecdsa-with-SHA256

30:46:02:21:00:97:5f:67:f2:59:88:cb:7d:37:d4:cc:38:f5:

38:08:42:1a:bc:ca:f9:25:48:8e:72:2c:c9:b2:03:53:36:84:

94:02:21:00:e1:e8:99:1b:49:ea:27:12:49:d4:25:f7:d4:bc:

f6:94:be:7e:f3:b7:fb:a3:a9:de:92:1d:0f:e4:0d:74:5e:88

B.3. Second Intermediate Certificate

-----BEGIN CERTIFICATE-----

```
MIIBoTCCAUAgaAwIBAgIEZ7N72zAKBggqhkhjOPQODAjaArMRUwEwYDVQQKDAxPcmdO
YW1lIEluYy4xEjAQBGNVBAMMCVVTQjoxYTBhOjAiGA8xOTcwMDEwMTAwMDAwMFoY
Dzk5OTkxMjMxMjM1OTU5WjBKMRUwEwYDVQQKDAxPcmdOYW1lIEluYy4xGTAXBgNV
BAsMEEV4YW1wbGUgRG12aXNpb24xYjAUBGNVBAMMDVVTQjoxYTBhOjAxMDEwWTAT
BgcqhkhjOPQIBBgqhkhjOPQMBBwNCAATluUu/+04EjUhtgUXJiwlodGNg7+4/34ir
T2uTCWUvSnEr7+pm0BHwnjjqWE6Qo/NCMPYoyD255uqvQULL9aesozUwMzAPBgNV
HRMBAf8EBTADAQH/MASGA1UdDwQEAwIBBjATBgNVHSUBAf8ECTAHBgVngREBATAK
BggqhkhjOPQODAgnJADBGAiEA7uwemS/h2W2EBVu9fpA3sx0KPvbS31RALQbjbqFQ
aokCIQDpJlzdGQSu0Fyr/qdkgCaoiglnN5aSRDvui9g96IspDg==
```

-----END CERTIFICATE-----

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 1739815899 (0x67b37bdb)

Signature Algorithm: ecdsa-with-SHA256

Issuer: O=OrgName Inc., CN=USB:1a0a:

Validity

Not Before: Jan 1 00:00:00 1970 GMT

Not After : Dec 31 23:59:59 9999 GMT

Subject: O=OrgName Inc., OU=Example Division, CN=USB:1a0a:0101

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:f5:b9:4b:bf:fb:4e:04:8d:48:6d:81:45:c9:8b:

0d:68:74:63:60:ef:ee:3f:df:88:ab:4f:6b:93:09:

65:2f:4a:71:2b:ef:ea:66:d0:11:d6:9e:38:ea:c0:

4e:90:a3:f3:42:30:f6:28:c8:3d:b9:e6:ea:af:41:

42:cb:f5:a7:ac

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Key Usage:

Certificate Sign, CRL Sign

X509v3 Extended Key Usage: critical

2.23.145.1.1

Signature Algorithm: ecdsa-with-SHA256

30:46:02:21:00:ee:ec:1e:99:2f:e1:d9:6d:84:05:5b:bd:7e:

90:37:b3:1d:0a:3e:f6:d2:df:54:40:2d:06:e3:6e:a1:50:6a:

89:02:21:00:e9:27:5c:c3:19:04:ae:d0:5c:ab:fe:a7:64:80:

26:a8:8a:0d:67:37:96:92:44:3b:ee:8b:d8:3d:e8:8b:29:0e

B.4. Leaf Certificate

-----BEGIN CERTIFICATE-----

```
MIICZjCCAg2gAwIBAgIJAesj2WQkFXXxMAoGCCqGSM49BAMCMEoxFTATBgNVBAoM
DE9yZ05hbWUgSW5jLjEzMBCGA1UECwwQRXhhbXBsZSBEaXZpc2lvdjEWMBCGA1UE
AwwNVVNCOjFhMGE6MDEwMTAiGA8xOTcwMDEwMTAwMDAwMFoYDzk5OTkxMjMxMjM1
OTU5WjBeMRIwEAYDVQQQtAwKA6yPZZCQVdfExFTATBgNVBAoMDE9yZ05hbWUgSW5j
LjEzMBCGA1UECwwQRXhhbXBsZSBEaXZpc2lvdjEWMBCGA1UEAwwNVVNCOjFhMGE6
MDEwMTBZBMGBYqGSM49AgEGCCqGSM49AwEHA0IABN38xqfvngwisG47yi6bm5VW
OpK6xJHCf3BvLpJozkdUlqPZZYZboVQW0NbdwM4IvvqAEM+mmXVvm++RVmyQpCyj
gcMwgcAwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCB4AwEwYDVR0lAQH/BAkwBwYF
Z4ERAQEwgY0GBWeBEQECBIGDBIGAIYV5y4cSv6OA0g437+SRFPT+42PnBTKLRTww
QwPFPaz7KCApDRXXST24hsyIk7CiXpRFkaJIHuoylbi+W1c5SdwfsMD9GmkUb7Ea
zCfR5nkBXyAnWuedkl5smoYN1Q312Qb0q6krXR07cMwAY4VWCnZQ1UDcsEaPsmLE
xB9DLyowCgYIKoZIZj0EAwIDRwAwRAIgZ3C/1XLN18MI+P5zJywQqR08VI0RhWgG
h0D72j+yPacCIC6Plwig431e/5QKM346Ra3OOFK4QmM5c5rGbYmu9CS5
```

-----END CERTIFICATE-----

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

01:eb:23:d9:64:24:15:75:f1

Signature Algorithm: ecdsa-with-SHA256

Issuer: O=OrgName Inc., OU=Example Division, CN=USB:1a0a:0101

Validity

Not Before: Jan 1 00:00:00 1970 GMT

Not After : Dec 31 23:59:59 9999 GMT

Subject: x500UniqueIdentifier=eb:23:d9:64:24:15:75:f1,
O=OrgName Inc., OU=Example Division, CN=USB:1a0a:0101

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

```
04:dd:fc:c6:a7:ef:9e:0c:22:b0:6e:3b:ca:2e:9b:
9b:95:56:3a:92:ba:c4:91:c2:7f:70:6f:2e:92:68:
ce:47:54:96:a3:d9:65:8c:db:a1:54:16:d0:d6:dd:
c0:ce:08:be:fa:80:10:cf:a6:99:75:6f:9b:ef:91:
56:6c:90:a4:2c
```

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Basic Constraints: critical

CA:FALSE

X509v3 Key Usage:

Digital Signature

X509v3 Extended Key Usage: critical

2.23.145.1.1

2.23.145.1.2:

{NOTE: ACD goes here but is omitted for this example}

Signature Algorithm: ecdsa-with-SHA256

```
30:44:02:20:67:70:bf:d5:72:cd:97:c3:08:f8:fe:73:27:2c:
10:a9:1d:3c:54:8d:11:85:68:06:87:40:fb:da:3f:b2:3d:a7:
02:20:2e:8f:d7:08:a0:e3:7d:5e:ff:94:0a:33:7e:3a:45:ad:
ce:38:52:b8:42:63:39:73:9a:c6:6d:89:ae:f4:24:b9
```

C Potential Attack Vectors

A list with examples of possible attacks against a Product is provided below. This list should be used as a checklist to determine whether a product has been thoroughly designed so that it can withstand known attacks that are most likely employed by common attackers. The list is partly based on the “Joint Interpretation Library - Application of Attack Potential to Smartcards” document and should be updated and reviewed regularly. A common criteria certificate with EAL5 and resistance against attackers with high attack potential determines that resistance against the attacks listed below has been achieved.

- Conformance testing of implemented algorithms (ECDSA, SHA256) according to test vectors and procedures described by NIST's Cryptographic Algorithm Validation Program (CAVP). See <http://csrc.nist.gov/groups/STM/cavp/index.html>
- Protection of secret key operations against timing analysis.
- Protection (e.g., randomization/masking) against standard analysis of power consumption (SPA/DPA) and electromagnetic emanation (SEMA/DEMA) of secret key operations.
- Protection against advanced side-channel attacks against ECC-ECDSA computation (e.g., refined power analysis, zero value attacks, address-bit DPA, template attacks) of secret key operations.
- Protection critical computations against fault insertion (temperature, voltage, frequency variation; spikes and glitches; light; forcing; radiation) and advanced fault attacks (e.g., DFA, multi-bit faults).
- Protection of secret key in non-volatile memory against extraction (e.g., memory encryption) and manipulation/modification using non-invasive, semi-invasive, or invasive attacks.
- Protection against probing or forcing of intermediate values of the secret key during transfer on a chip internal bus.
- Protection and post-production lock down of test modes (e.g., JTAG) and scan chain.
- Protection against advanced invasive attacks like micro-probing or modification of circuits using a focused ion beam (FIB).
- Test of the statistical properties of the device-internal true random number generator (TRNG)
- Online test to recognize failure or manipulation by an attacker of the TRNG during operation.
- Secure system reset in case of the detection of an attack.
- Protection and thorough testing (e.g., fuzzing) to prevent (logical) attacks on software (e.g., bugs) like buffer overflows, man-in-the-middle, replay attacks, undocumented commands, bypass of authentication or access control.