Universal Serial Bus Type-C™ Authentication Specification

Revision 1.0 with ECN and Errata through July 24, 2017

CONTENTS

TABLES

FIGURES

Specification Work Group Chairs / Specification Editors

Renesas Electronics Corp.	Co-Chair	Bob Dunstan
Intel Corporation	Co-Chair	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 Kuhnis	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 Borve
	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
	Dan Aoki		

ROHM Co., Ltd.	Ruben Balbuena	Kris Bahar	Nobutaka Itakura
	Takashi Sato		
Samsung Electronics Co., Ltd.	Tong Kim	Jagoun Koo	Soondo Kim
SiliConch	Rakesh Polasa	Jaswanth Ammineni	Kaustubh Kumar
	Pavitra Balasubramanian	Aniket Mathad	Shubham Paliwal
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		

Revision History

Revision	Date	Description
1.0	March 25, 2016	Initial Release
1.0 + ECN and Errata	February 2, 2017	Includes ECN and errata through February 2, 2017
1.0 + ECN and Errata	July 24, 2017	Includes ECN and errata through July 24, 2017

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-S!!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)
  - ISO/IEC 15408 Evaluation criteria for IT security Parts 1-3
- ***ECDSA***:
  - ANSI X9.62; NIST-FIPS-186-4, Section 6 (available at: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.)
  - ISO/IEC 14888-3 Digital signatures with appendix -- Part 3: Discrete logarithm based mechanisms (Clause 6.6)
- ***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.)
  - ISO/IEC 15946 Cryptographic techniques based on elliptic curves (NIST P-256 is included as example)
    - *Notes: ISO/IEC 15946 series treat elliptic curves differently from FIPS 186-4. ISO/IEC 15946-5 is about elliptic curve generation. That is, based on the*

method in part 5, each application and implementation can generate its own curves to use. In other words, no ISO/IEC recommended curves. P-256 is consider an example in ISO/IEC 15946. Note that Elliptic Curve signatures and key establishment schemes have been moved to ISO/IEC 14888 and ISO/IEC 11770 respectively together with other discrete log based mechanisms. Test vectors (examples) use P-256 are included for each parts for those mechanisms.

- ***SHA256***:
  - NIST-FIPS-180-4 (available at: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.)
  - ISO/IEC 10118-3 Hash-functions -- Part 3: Dedicated hash-functions (Clause 10)
- ***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.)
    - *Note: NIST-SP-800-90A was withdrawn June 2015 and replaced by NIST-SP-800-90A Revision 1 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.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 is still in DRAFT phase*.

[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.
TLV	Type, Length, Value
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 defi ned 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

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

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.

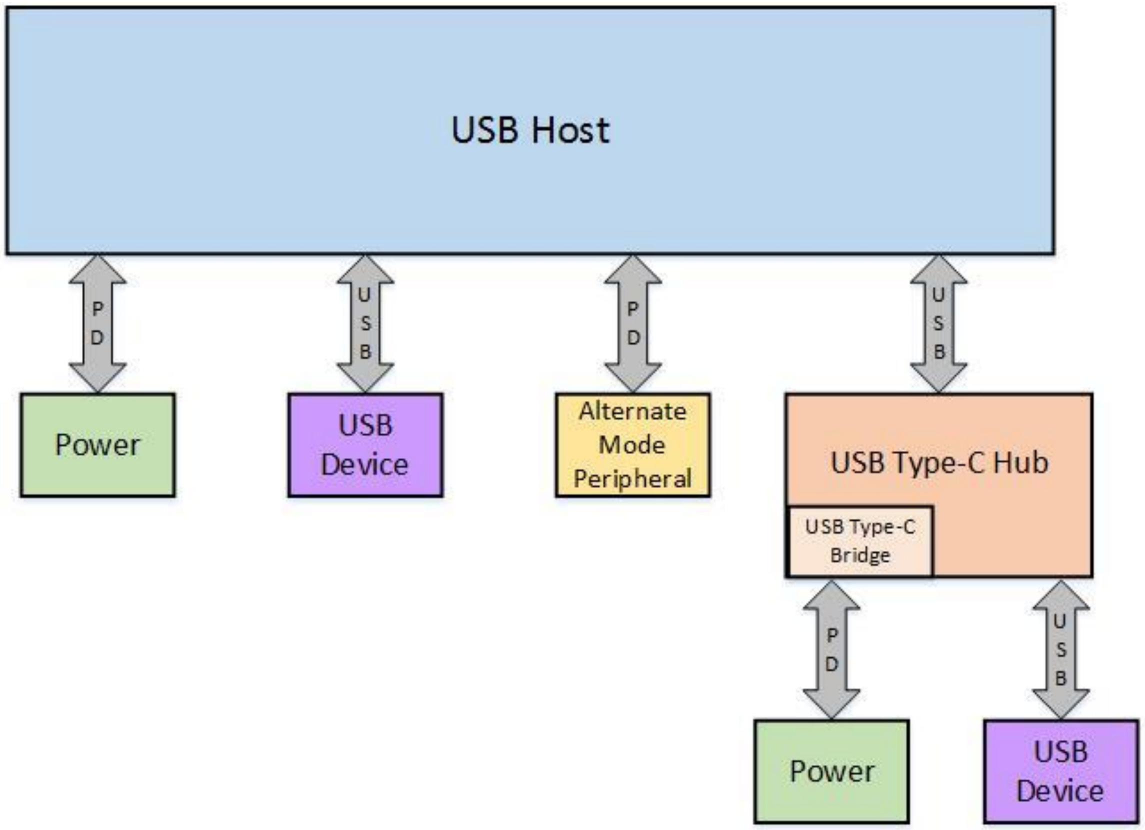
2 Overview

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
X509v3, ***DER*** encoding	Certificate format
ECDSA using the ***NIST P256, secp256r1*** curve, uncompressed point format	Digital signing of Certificates and Authentication Messages
SHA256	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 ***USBTYPEC 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.

Certificates and the fields, attributes, and extensions defined therein are Big Endian.

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. If present, it shall only be present in a Leaf certificate. 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 should 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). Use of the recommended *notBefore* and *notAfter* values will maximize compatibility with certificate processing stacks.

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 contain 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	*Length*	2	Total length of Certificate Chain in bytes including all fields in this table This field is little endian.
2	*Reserved*	2	Set to zero
4	*RootHash*	32	32-byte ***SHA256*** hash of the Root Certificate. *Note that Root Certificate is **ASN.1 DER**-encoded for hash.* This field is big endian.
36	*Certificates*	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. This field is big endian.

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 Certificate Chain 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. Private keys shall be generated, provisioned, and stored in a manner that adequately protects the confidentiality of the key.

The following rules govern private key uniqueness:

- A private key used by one Product shall not be used by any other Products.
- A private key used by a Product in one slot shall not be used in any other slots.

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 6.3 (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, upon receiving an Authentication Request, 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. Neither an Authentication Initiator nor an Authentication Responder shall add any padding after an Authentication Message. 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	*ProtocolVersion*	1	Section 5.1.1
1	*MessageType*	1	Section 5.1.2
2	*Param1*	1	Section 5.1.3
3	*Param2*	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 by an Authentication Initiator to send a command to an Authentication Responder and/or retrieve data. Authentication Request types are listed in Table 5-3.

An Authentication Initiator shall not send another Authentication Request until it has either received a response for or timed out the previously sent Authentication Request.

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	*ProtocolVersion*	1	V1.0
1	*MessageType*	1	GET_DIGESTS
2	*Param1*	1	Reserved
3	*Param2*	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	*ProtocolVersion*	1	V1.0
1	*MessageType*	1	GET_CERTIFICATE
2	*Param1*	1	Slot number of the target Certificate Chain to read from. The value in this field shall be between 0 and 7 inclusive.
3	*Param2*	1	Reserved

Table 5-6: GET_CERTIFICATE Request Payload

Offset	Field	Size	Value
4	*Offset*	2	Offset in bytes from the start of the Certificate Chain to where the read request begins. This field is little endian.
6	*Length*	2	Length in bytes of the read request. This field is little endian.

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	*ProtocolVersion*	1	V1.0
1	*MessageType*	1	CHALLENGE
2	*Param1*	1	Slot number of the recipient's Certificate Chain that will be used for Authentication
3	*Param2*	1	Reserved

Table 5-8: CHALLENGE Request Payload

Offset	Field	Size	Description
4	*Nonce*	32	Random 32-byte nonce chosen by the Authentication Initiator.

5.3 Authentication Responses

Authentication Responses are used by an Authentication Responder to respond to an Authentication Request. 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	*ProtocolVersion*	1	V1.0
1	*MessageType*	1	DIGESTS
2	*Param1*	1	Capabilities Field; shall be set to 01h for this specification. All other values reserved.
3	*Param2*	1	Slot mask. The bit in position K of this byte shall be set to 1b if and only if slot number K contains a Certificate Chain for the protocol version in the *ProtocolVersion* field. (Bit 0 is the least significant bit of the byte.) The number of digests returned shall be equal to the number of bits set in this byte. The digests shall be returned in order of increasing slot number.

Table 5-11: DIGESTS Response Payload

Offset	Field	Size	Value
4	*Digest[0]*	32	32-byte SHA-256 digest of the first Certificate Chain. This field is big endian.
…	…	…	…
4 + (32 * (n -1))	*Digest[n-1]*	32	32-byte SHA-256 digest of the last (nth) Certificate Chain. This field is big endian.

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	*ProtocolVersion*	1	V1.0
1	*MessageType*	1	CERTIFICATE
2	*Param1*	1	Slot number of the Certificate Chain returned
3	*Param2*	1	Reserved

Table 5-13: CERTIFICATE Response Payload

Offset	Field	Size	Value	Description
4	*CertChain*	*Length*	Data	Requested contents of target Certificate Chain. The endianness for a Certificate Chain is defined in Table 3-1.

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	*ProtocolVersion*	1	V1.0
1	*MessageType*	1	CHALLENGE_AUTH
2	*Param1*	1	Shall contain the Slot number in the *Param1* field of the corresponding CHALLENGE Request
3	*Param2*	1	Slot mask. The bit in position K of this byte shall be set to 1b if and only if slot number K contains a Certificate Chain for the protocol version in the *ProtocolVersion* field. (Bit 0 is the least significant bit of the byte.)

Table 5-15: CHALLENGE_AUTH Response Payload

Offset	Field	Size	Value
4	*MinProtocolVersion*	1	Minimum protocol version supported by this Device
5	*MaxProtocolVersion*	1	Maximum protocol version supported by this Device
6	*Capabilities*	1	Set to 01h for this specification. All other values reserved
7	*Reserved*	1	Reserved
8	*CertChainHash*	32	32-byte ***SHA256*** hash of the Certificate Chain used for Authentication. This field is big endian.
40	*Salt*	32	32-byte value chosen by the Authentication Responder. *Note: the Salt can be random, fixed, or any other value*
72	*Context Hash*	32	See Section 6.4 for PD Products or Section 7.5 for USB Products. This field is big endian.
104	*Signature*	64	See Section 5.3.3.1. This field is little endian.

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	*ReqMsg*	36	Full contents (i.e. header and payload) of the corresponding CHALLENGE Request
36	*RespMsg*	104	Contents of the CHALLENGE_AUTH Response being signed excluding the *Signature* 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	*ProtocolVersion*	1	V1.0[1]
1	*MessageType*	1	ERROR
2	*Param1*	1	Error Code. See Table 5-18.
3	*Param2*	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

[1]*Note: 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).

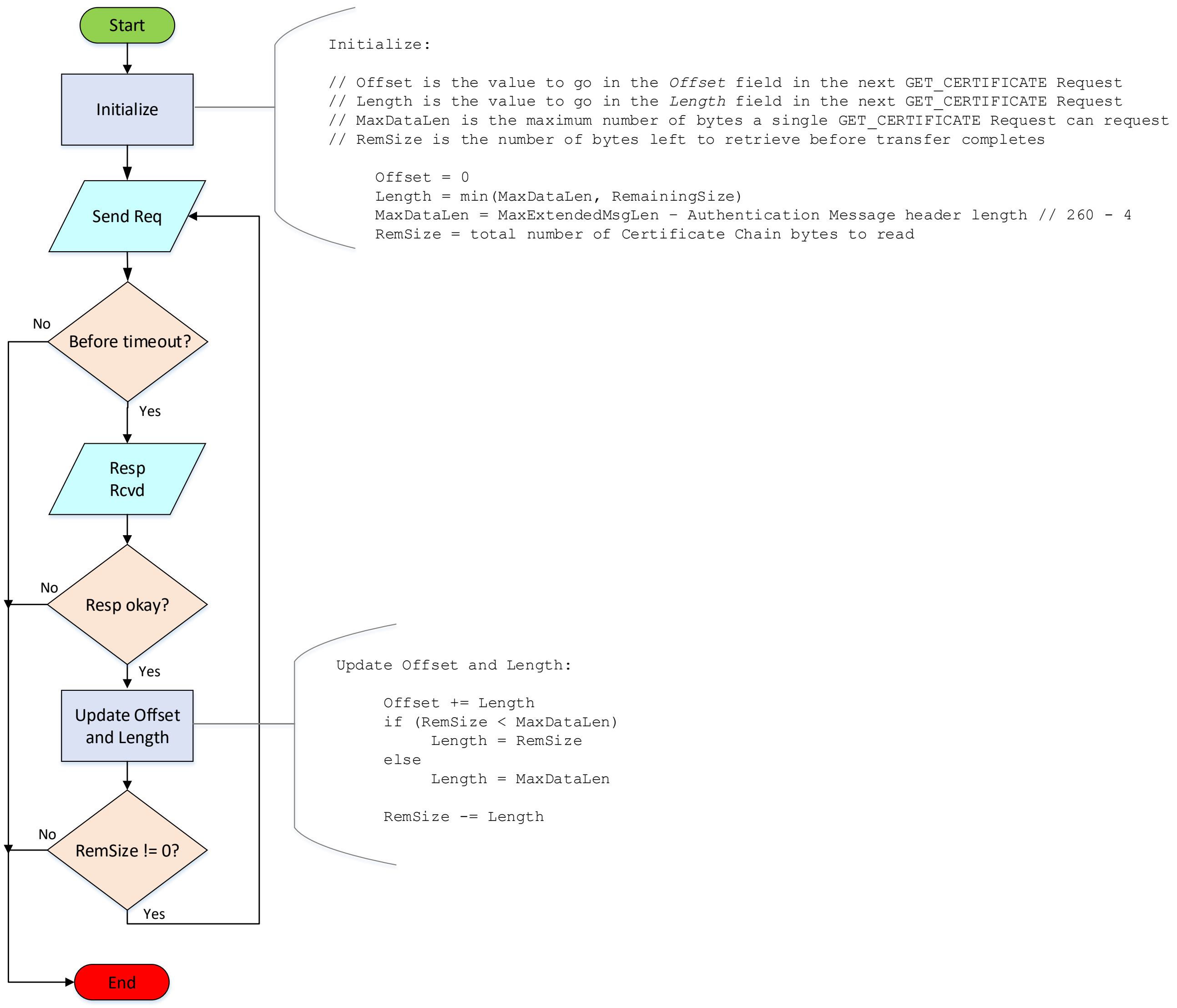
6.2 Transfers greater than *MaxExtendedMsgLen*

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 shall break 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. Each Authentication Response is sent as in the SRPDB of a Security_Response PD Message.

Figure 6-1 shows an example of how an Authentication Initiator can break up and complete a security transfer when more than *MaxExtendedMsgLen* number of bytes are read from the Certificate Chain of an Authentication Responder.

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



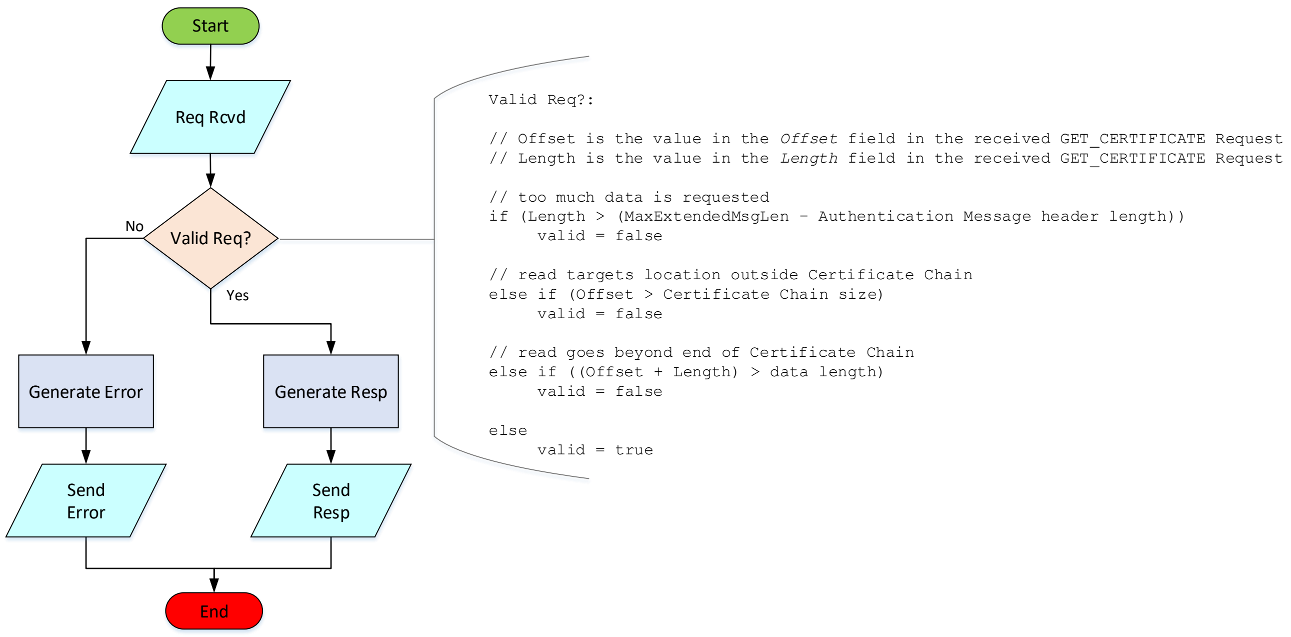
```
Initialize:

// Offset is the value to go in the Offset field in the next GET_CERTIFICATE Request
// Length is the value to go in the Length field in the next GET_CERTIFICATE Request
// MaxDataLen is the maximum number of bytes a single GET_CERTIFICATE Request can request
// RemSize is the number of bytes left to retrieve before transfer completes

     Offset = 0
     Length = min(MaxDataLen, RemainingSize)
     MaxDataLen = MaxExtendedMsgLen - Authentication Message header length // 260 - 4
     RemSize = total number of Certificate Chain bytes to read
```

```
Update Offset and Length:

     Offset += Length
     if (RemSize < MaxDataLen)
          Length = RemSize
     else
          Length = MaxDataLen

     RemSize -= Length
```

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

Figure 6-2 is an example of how an Authentication Responder can respond to a GET_CERTIFICATE Request. The criteria the Authentication Responder uses to determine whether or not a given GET_CERTIFICATE Request is valid is listed in the pseudocode at the right of the figure.

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

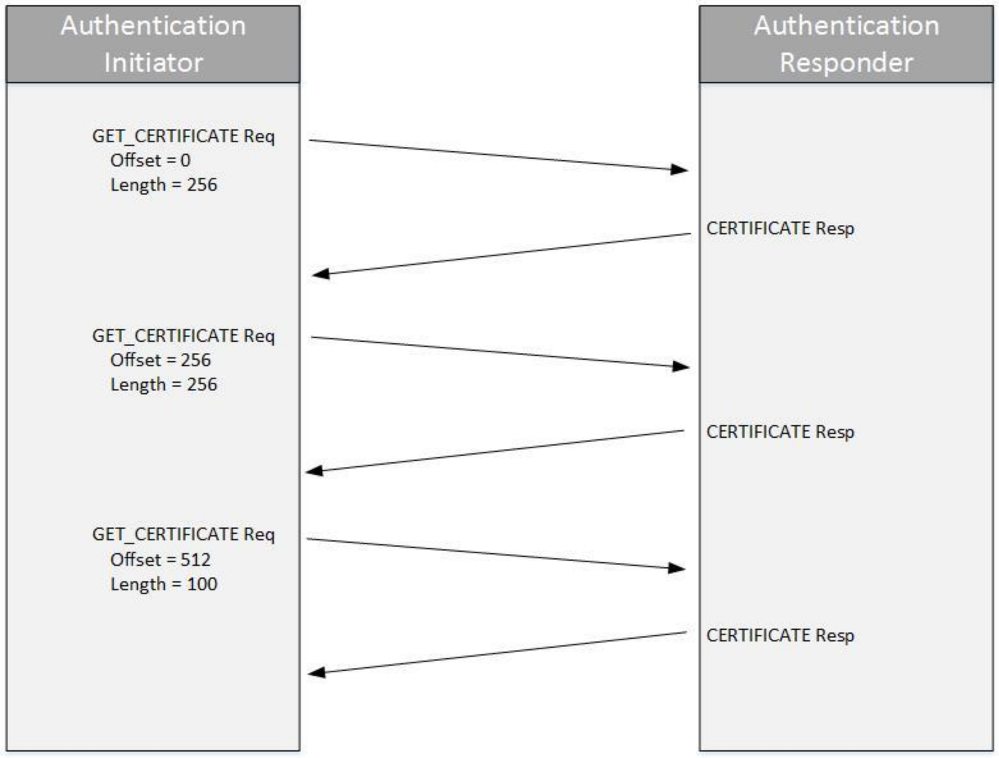


```
Valid Req?:

// Offset is the value in the Offset field in the received GET_CERTIFICATE Request
// Length is the value in the Length field in the received GET_CERTIFICATE Request

// too much data is requested
if (Length > (MaxExtendedMsgLen - Authentication Message header length))
     valid = false

// read targets location outside Certificate Chain
else if (Offset > Certificate Chain size)
     valid = false

// read goes beyond end of Certificate Chain
else if ((Offset + Length) > data length)
     valid = false

else
     valid = true
```

Figure 6-3 shows an example of a GET_CERTIFICATE Request for a 612-Byte Certificate Chain using the processes in

Figure 6-1 and Figure 6-2.

Figure 6-3 Example 612-Byte Certificate Chain Read



6.3 Timing Requirements for PD Security Extended Messages

6.3.1 Authentication Initiator

Table 6-1 shows the timeout values that apply to an Authentication Initiator. The timeout values shall be measured from when the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the Authentication Request Message, has been received by the PD Physical Layer until the last bit of the Authentication Response Message ***EOP*** has been received by the PD Physical Layer.

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

Parameter	Timeout Value		Description
	Unchunked	Chunked	
tDigestRcvd	40 ms	200 ms	Timeout for a GET_DIGESTS Authentication Request.
tCertRcvd	40 ms	200 ms	Timeout for a GET_CERTIFICATE Authentication Request.
tChallengeAuthRcvd	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 Initiator does not receive an Authentication Response within *tChallengeAuthRcvd* of sending a CHALLENGE Authentication Request, it is considered an error. Subsequent handling of this error is outside the scope of this specification.

6.3.2 Authentication Responder

Table 6-2 gives timing requirements that an Authentication Responder shall meet. The timing requirements shall be measured from when the last bit of the Authentication Request Message ***EOP*** has been received by the PD Physical Layer until the last bit of the ***GoodCRC*** Message ***EOP*** corresponding to the Authentication Response Message has been received by the PD Physical Layer.

Table 6-2: Timing Requirements for PD Authentication Responder

Parameter	**Timing Value**		**Description**
	Unchunked	**Chunked**	
tDigestSent	30 ms	135 ms	Maximum time between receiving a GET_DIGESTS Authentication Request and sending an Authentication Response.
tCertSent	30 ms	135 ms	Maximum time between receiving a GET_CERTIFICATE Authentication Request and sending an Authentication Response.
tChallengeAuthSent	530 ms	635 ms	Maximum time between receiving a CHALLENGE Authentication Request and sending an Authentication Response.

An Authentication Response shall be sent within *tDigestSent* of receiving a GET_DIGESTS Authentication Request.

An Authentication Response shall be sent within *tCertSent* of receiving a GET_CERTIFICATE Authentication Request.

An Authentication Response shall be sent within *tChallengeAuthSent* of receiving a CHALLENGE Authentication Request.

6.4 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	*bLength*	1	Number	Size of this Descriptor in Bytes
1	*bDescriptorType*	1	Constant	Descriptor type: DEVICE CAPABILITY (16)
2	*bDevCapabilityType*	1	Number	AUTHENTICATION
3	*bmAttributes*	1	Bitmap	Bitmap encoding of supported features. Bit | Description 0 | Shall be set to 1 to indicate that firmware can be updated. Otherwise, shall be set to zero. 1 | Shall be set to 1 to indicate that Device changes interfaces when updated. Otherwise, shall be set to zero. 7:2 | Reserved. Shall be set to zero.
4	*bcdProtocolVersion*	1	Number	Shall be set to the USB Type-C Authentication Protocol Version
5	*bcdCapability*	1	Number	Shall be set to the same value as *Param1* 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 bRequest	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	*ProtocolVersion*	1	Number	Maps to high byte of wValue
1	*MessageType*	1	Number	Maps to low byte of wValue
2	*Param1*	1	Number	Maps to high byte of wIndex
3	*Param2*	1	Number	Maps to low byte of wIndex

A USB Device shall respond with a Request Error if *wLength* for a particular Response type does not match the values set forth in this section. The behavior of a USB Device is not specified if *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	Request is invalid and Device shall respond with a Request Error.
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.

A USB Device shall respond with a Request Error if *wLength* for a particular Response type does not match the values set forth in this section. The behavior of a USB Device is not specified if *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	Request is invalid and Device shall respond with a Request Error.
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	DIGESTS 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	168	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***. Additional timing requirements are defined in Section 7.4.1 for USB Hosts and Section 7.4.2 for USB Devices.

7.4.1 USB Host Timing Requirements

A USB Host shall use the timeout values in Table 7-12.

Table 7-12: Authentication Initiator Timeout Values

Parameter	Timeout Value	Description
tDigestIN	100 ms	Timeout for a GET_DIGESTS Authentication Request.
tCertOUT	100 ms	Timeout for an AUTH_OUT control transfer carrying a GET_CERTIFICATE Authentication Request.
tCertIN	500 ms	Timeout for an AUTH_IN control transfer carrying a CERTIFICATE Authentication Response.
tChallengeOUT	100 ms	Timeout for an AUTH_OUT control transfer carrying a CHALLENGE Authentication Request.
tChallengeIN	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.

7.4.2 USB Device Timing Requirements

A USB Device shall use the response times in Table 7-13.

Table 7-13: Authentication Responder Response Times

Parameter	Maximum Response Time	Description
tDigestSent	95 ms	Maximum time between receiving a DIGESTS Authentication Request and sending an Authentication Response.
tCertACK	95 ms	Maximum time between receiving a GET_CERTIFICATE Request and sending an ACK.
tCertSent	495 ms	Maximum time between receiving a CERTIFICATE Authentication Request and sending an Authentication Response.
tChallengeACK	95 ms	Maximum time between receiving a CHALLENGE Request and sending an ACK.
tChallengeAuthSent	595 ms	Maximum time between receiving a CHALLENGE_AUTH Authentication Request and sending an Authentication Response.

A USB Device shall respond to an Authentication Initiator within *tDigestSent* of receiving an AUTH_IN control transfer carrying a GET_DIGESTS Authentication Request.

A USB Device shall ACK an AUTH_OUT control transfer carrying a GET_CERTIFICATE Request within *tCertACK* of receiving it.

A USB Device shall respond to an Authentication Initiator within *tCertSent* of receiving an AUTH_IN control transfer carrying a CERTIFICATE Authentication Request.

A USB Device shall ACK an AUTH_OUT control transfer carrying a CHALLENGE Request within *tChallengeACK* of receiving it.

A USB Device shall respond to an Authentication Initiator within *tChallengeAuthSent* of receiving an AUTH_IN control transfer carrying a CHALLENGE_AUTH Authentication Request.

7.5 Context Hash

This field shall contain 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
MaxLeafCertSize	640 bytes
MaxIntermediateCertSize	512 bytes
MaxACDSize	128 bytes
MaxCertChainSize	4096 bytes

A ACD

A.1. ACD Formatting

ACD formatting consists of a sequence of zero or more Type, Length, Value (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.

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

All TLVs are big-endian (i.e. MSB to LSB).

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

* 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	CABLE_CAPABILITIES
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-2.

Figure A-2: Bitmap of the Common Criteria Identifier

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

* 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-3.

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	Length	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 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	Length	See Table A-23

Table A-23: Vendor Extension TLV Data

Offset	Field	Size	Description
0	VID	2	Vendor ID of vendor defining the Data field
2	Vendor Defined	(Length – 2)	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-24: Extension TLV Fields

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

A.2. ACD for a PD Product

The ACD for a PD 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.

Table A-25: PD Product ACD TLVs

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

A.3. ACD for a USB Product

The ACD for a USB Product is defined in Table A-26. 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.

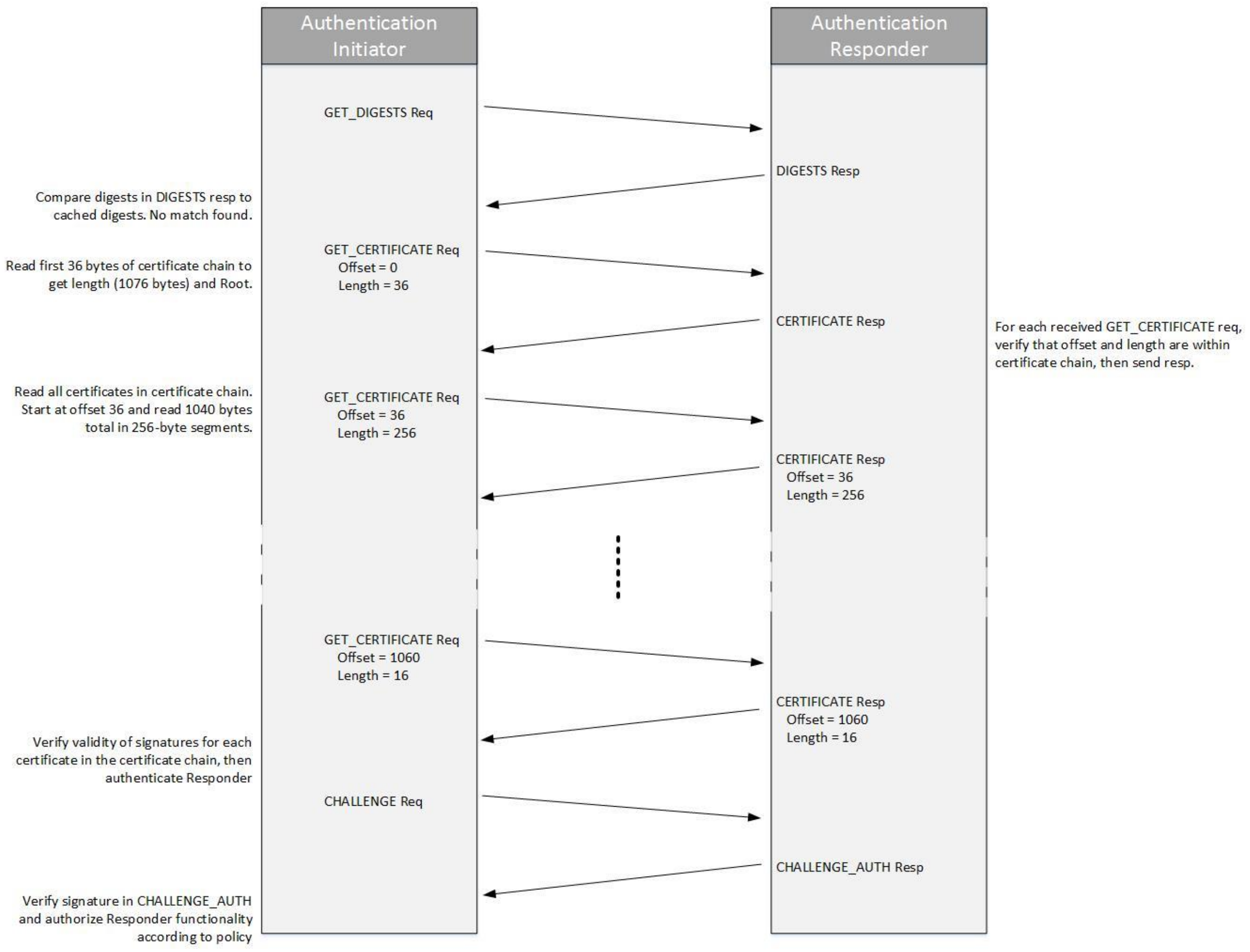
Table A-26: USB Product ACD TLVs

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 Cryptographic Examples

The examples in this Appendix are included to help illustrate the concepts defined in the specification and are informative only.

B.1. Example Authentication Sequence


Authentication Initiator
Authentication Responder
GET_DIGESTS Req
DIGESTS Resp
Compare digests in DIGESTS resp to cached digests. No match found.
GET_CERTIFICATE Req Offset = 0 Length = 36
Read first 36 bytes of certificate chain to get length (1076 bytes) and Root.
CERTIFICATE Resp
For each received GET_CERTIFICATE req, verify that offset and length are within certificate chain, then send resp.
Read all certificates in certificate chain. Start at offset 36 and read 1040 bytes total in 256-byte segments.
GET_CERTIFICATE Req Offset = 36 Length = 256
CERTIFICATE Resp Offset = 36 Length = 256
GET_CERTIFICATE Req Offset = 1060 Length = 16
CERTIFICATE Resp Offset = 1060 Length = 16
Verify validity of signatures for each certificate in the certificate chain, then authenticate Responder
CHALLENGE Req
CHALLENGE_AUTH Resp
Verify signature in CHALLENGE_AUTH and authorize Responder functionality according to policy

B.2. Example Certificate Chain Topology

B.2.1. Certificate Chain

```
87 03 00 00 EB 13 EB C1 8D F6 73 03 9B 76 99 66
AD A3 E5 26 AC 40 77 09 C2 37 24 FB E0 C7 B2 E0
02 30 FF 69 30 82 01 80 30 82 01 25 A0 03 02 01
02 02 09 00 83 74 6A 3F AE 6A DF 28 30 0A 06 08
2A 86 48 CE 3D 04 03 02 30 24 31 12 30 10 06 03
55 04 0A 0C 09 55 53 42 2D 49 46 20 43 41 31 0E
30 0C 06 03 55 04 03 0C 05 55 53 42 3A 3A 30 22
18 0F 31 39 37 30 30 31 30 31 30 30 30 30 30 30
5A 18 0F 39 39 39 39 31 32 33 31 32 33 35 39 35
39 5A 30 2B 31 15 30 13 06 03 55 04 0A 0C 0C 4F
72 67 4E 61 6D 65 20 49 6E 63 2E 31 12 30 10 06
03 55 04 03 0C 09 55 53 42 3A 31 61 30 61 3A 30
59 30 13 06 07 2A 86 48 CE 3D 02 01 06 08 2A 86
48 CE 3D 03 01 07 03 42 00 04 3F 11 11 AC 9A A1
C2 43 8E 66 D8 41 C1 4F 87 6A DF 35 0E AA 73 6C
89 44 5F 42 19 B3 5C F8 F3 37 C8 9D 64 EC 65 B1
16 4D C7 E6 89 60 62 6A D5 D5 AD 37 21 D3 60 39
38 1B 2F 01 4B 43 09 4C 85 B1 A3 35 30 33 30 0F
06 03 55 1D 13 01 01 FF 04 05 30 03 01 01 FF 30
0B 06 03 55 1D 0F 04 04 03 02 01 06 30 13 06 03
55 1D 25 01 01 FF 04 09 30 07 06 05 67 81 11 01
01 30 0A 06 08 2A 86 48 CE 3D 04 03 02 03 49 00
30 46 02 21 00 9D F3 20 EC 82 63 F4 D0 74 1E D5
45 A0 B8 D5 FA 07 59 42 88 DE 82 BA 46 A9 D0 91
09 41 CC 6D A6 02 21 00 C3 CC 17 0C B5 AF 5D F3
A5 53 81 D4 7D 7B 06 AD A3 0A A7 93 1C A7 B0 11
6A D2 0D 3C 16 94 95 7B 30 82 01 DB 30 82 01 81
A0 03 02 01 02 02 09 00 81 9F 59 97 6B 38 47 91
30 0A 06 08 2A 86 48 CE 3D 04 03 02 30 2B 31 15
30 13 06 03 55 04 0A 0C 0C 4F 72 67 4E 61 6D 65
20 49 6E 63 2E 31 12 30 10 06 03 55 04 03 0C 09
55 53 42 3A 31 61 30 61 3A 30 22 18 0F 31 39 37
30 30 31 30 31 30 30 30 30 30 30 5A 18 0F 39 39
39 39 31 32 33 31 32 33 35 39 35 39 5A 30 42 31
15 30 13 06 03 55 04 0A 0C 0C 4F 72 67 4E 61 6D
65 20 49 6E 63 2E 31 16 30 14 06 03 55 04 03 0C
0D 55 53 42 3A 31 61 30 61 3A 30 31 30 31 31 11
30 0F 06 03 55 04 05 13 08 35 35 36 36 37 37 38
38 30 59 30 13 06 07 2A 86 48 CE 3D 02 01 06 08
2A 86 48 CE 3D 03 01 07 03 42 00 04 A9 B0 F8 66
B0 2D 91 2B 87 64 22 57 D2 AE 0B 07 E1 FA 83 A6
8E B4 4F 3F 16 79 43 A3 E5 D8 00 72 DC 0A D5 B3
00 A4 FB 8B C0 05 3B 4D 7C 9D 8D 48 BB AC 46 8C
E5 28 B7 5B 1C 5A FA 2E 4D DA 38 45 A3 73 30 71
30 0C 06 03 55 1D 13 01 01 FF 04 02 30 00 30 0B
06 03 55 1D 0F 04 04 03 02 07 80 30 13 06 03 55
1D 25 01 01 FF 04 09 30 07 06 05 67 81 11 01 01
30 3F 06 05 67 81 11 01 02 04 36 00 02 40 00 01
04 00 00 12 34 02 16 02 01 01 00 03 07 01 00 2A
0A 2A 0A 2A 0A 00 00 00 01 2A 01 91 2C 05 06 00
00 00 55 1A 0A FD 04 54 45 53 54 FE 04 1A 0A 12
34 30 0A 06 08 2A 86 48 CE 3D 04 03 02 03 48 00
30 45 02 20 73 DB 58 34 78 91 0C B9 C3 F5 6C 00
AB 6E 9C E0 E0 13 B1 A3 41 D4 12 5F 3B E0 E9 31
A3 C3 6E D5 02 21 00 C6 CF 1D BD 9D 0F B1 2C 5B
EF 17 4E 30 44 52 C3 27 FA A2 FA 40 29 F5 49 65
31 21 05 C2 BB 2B 35
```

B.2.1.1. Intermediate Certificate

```
-----BEGIN CERTIFICATE-----
MIIBgDCCASWgAwIBAgIJAIN0aj+uat8oMAoGCCqGSM49BAMCMCQxEjAQBgNVBAoM
CVVTQi1JRiBDQTEOMAwGA1UEAwwFVVNCOjowIhgPMTk3MDAxMDEwMDAwMDBaGA85
OTk5MTIzMTIzNTk1OVowKzEVMBMGA1UECgwMT3JnTmFtZSBJbmMuMRIwEAYDVQQD
DAlVU0I6MWEwYTowWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQ/ERGsmqHCQ45m
2EHBT4dq3zUOqnNsiURfQhmzXPjzN8idZOxlsRZNx+aJYGJq1dWtNyHTYDk4Gy8B
S0MJTIWxozUwMzAPBgNVHRMBAf8EBTADAQH/MAsGA1UdDwQEAwIBBjATBgNVHSUB
Af8ECTAHBgVngREBATAKBggqhkjOPQQDAgNJADBGAiEAnfMg7IJj9NB0HtVFoLjV
+gdZQojegrpGqdCRCUHMbaYCIQDDzBcMta9d86VTgdR9ewatowqnkxynsBFq0g08
FpSVew==
-----END CERTIFICATE-----
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            83:74:6a:3f:ae:6a:df:28
    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:
                    04:3f:11:11:ac:9a:a1:c2:43:8e:66:d8:41:c1:4f:
                    87:6a:df:35:0e:aa:73:6c:89:44:5f:42:19:b3:5c:
                    f8:f3:37:c8:9d:64:ec:65:b1:16:4d:c7:e6:89:60:
                    62:6a:d5:d5:ad:37:21:d3:60:39:38:1b:2f:01:4b:
                    43:09:4c:85:b1
                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
         30:46:02:21:00:9d:f3:20:ec:82:63:f4:d0:74:1e:d5:45:a0:
         b8:d5:fa:07:59:42:88:de:82:ba:46:a9:d0:91:09:41:cc:6d:
         a6:02:21:00:c3:cc:17:0c:b5:af:5d:f3:a5:53:81:d4:7d:7b:
         06:ad:a3:0a:a7:93:1c:a7:b0:11:6a:d2:0d:3c:16:94:95:7b
```

B.2.1.2. Leaf Certificate

```
-----BEGIN CERTIFICATE-----
MIIB2zCCAYGgAwIBAgIJAIGfWZdrOEeRMAoGCCqGSM49BAMCMCsxFTATBgNVBAoM
DE9yZ05hbWUgSW5jLjESMBAGA1UEAwwJVVNCOjFhMGE6MCIYDzE5NzAwMTAxMDAw
MDAwWhgPOTk5OTEyMzEyMzU5NTlaMEIxFTATBgNVBAoMDE9yZ05hbWUgSW5jLjEW
MBQGA1UEAwwNVVNCOjFhMGE6MDEwMTERMA8GA1UEBRMINTU2Njc3ODgwWTATBgcq
hkjOPQIBBggqhkjOPQMBBwNCAASpsPhmsC2RK4dkIlfSrgsH4fqDpo60Tz8WeUOj
5dgActwK1bMApPuLwAU7TXydjUi7rEaM5Si3Wxxa+i5N2jhFo3MwcTAMBgNVHRMB
Af8EAjAAMAsGA1UdDwQEAwIHgDATBgNVHSUBAf8ECTAHBgVngREBATA/BgVngREB
AgQ2AAJAAAEEAAASNAIWAgEBAAMHAQAqCioKKgoAAAABKgGRLAUGAAAAVRoK/QRU
RVNU/gQaChI0MAoGCCqGSM49BAMCA0gAMEUCIHPbWDR4kQy5w/VsAKtunODgE7Gj
QdQSXzvg6TGjw27VAiEAxs8dvZ0PsSxb7xdOMERSwyf6ovpAKfVJZTEhBcK7KzU=
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            81:9f:59:97:6b:38:47:91
    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.,
CN=USB:1a0a:0101/serialNumber=55667788
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:a9:b0:f8:66:b0:2d:91:2b:87:64:22:57:d2:ae:
                    0b:07:e1:fa:83:a6:8e:b4:4f:3f:16:79:43:a3:e5:
                    d8:00:72:dc:0a:d5:b3:00:a4:fb:8b:c0:05:3b:4d:
                    7c:9d:8d:48:bb:ac:46:8c:e5:28:b7:5b:1c:5a:fa:
                    2e:4d:da:38:45
                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:
                ..@......4..........*
*
*
....*..,.....U.
..TEST...
.4
    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:20:73:db:58:34:78:91:0c:b9:c3:f5:6c:00:ab:6e:
         9c:e0:e0:13:b1:a3:41:d4:12:5f:3b:e0:e9:31:a3:c3:6e:d5:
         02:21:00:c6:cf:1d:bd:9d:0f:b1:2c:5b:ef:17:4e:30:44:52:
         c3:27:fa:a2:fa:40:29:f5:49:65:31:21:05:c2:bb:2b:35
```

B.2.1.2.1. ACD

Table B-1: Version TLV Fields

Offset	Field	Size	Value
0	Type	1	00h (VERSION)
1	Length	1	02h
2	Data	2	4000h

Table B-2: XID TLV Fields

Offset	Field	Size	Value
0	Type	1	01h (XID)
1	Length	1	04h
2	Data	4	00001234h

Table B-3: Power Source Capabilities TLV Fields

Offset	Field	Size	Value
0	Type	1	02h (POWER_SOURCE_CAPABILITIES)
1	Length	1	16h
2	Data	22	02h (Version) 01h (FW Version) 01h (HW Version) 00h (Voltage Regulation) 03h (Hold Up time) 07h (Compliance) 01h (Touch Current) 00h (Reserved) 2A0Ah (Peak Current1) 2A0Ah (Peak Current2) 2A0Ah (Peak Current3) 00h (Touch Temp) 00h (Source Inputs) 00h (Batteries) 01h (Num PDOs) 2A01912Ch (PDOs)

Table B-4: Security Description TLV Fields

Offset	Field	Size	Value
0	Type	1	05h (SECURITY_DESCRIPTION)
1	Length	1	06h
2	Data	6	00h (FIPS/ISO Identifier) 0000h (Common Criteria Identifier) 55h (Security Analysis Identifier) 1A0Ah (IC Vendor Identifier)

Table B-5: Playpen TLV Fields

Offset	Field	Size	Value
0	Type	1	FDh (PLAYPEN)
1	Length	1	04h
2	Data	4	54455354h

Table B-6: Vendor Extension TLV Fields

Offset	Field	Size	Value
0	Type	1	FFh (VENDOR_EXTENSION)
1	Length	1	04h
2	Data	4	1A0Ah (VID) 1234h (Vendor Defined)

B.2.2. Root Certificate

```
-----BEGIN CERTIFICATE-----
MIIBcDCCARagAwIBAgIBATAKBggqhkjOPQQDAjAkMRIwEAYDVQQKDAlVU0ItSUYg
Q0ExDjAMBgNVBAMMBVVTQjo6MCIYDzE5NzAwMTAxMDAwMDAwWhgPOTk5OTEyMzEy
MzU5NTlaMCQxEjAQBgNVBAoMCVVTQi1JRiBDQTEOMAwGA1UEAwwFVVNCOjowWTAT
BgcqhkjOPQIBBggqhkjOPQMBBwNCAAQWcYqkNu02Tci2l7m7go+whORaE+ziCWBe
6+YM85dWe4H5CtSHPcwetDJEzpl6m6O3+w/lNgIgAyIWddEj0sMVozUwMzAPBgNV
HRMBAf8EBTADAQH/MAsGA1UdDwQEAwIBBjATBgNVHSUBAf8ECTAHBgVngREBATAK
BggqhkjOPQQDAgNIADBFAiAQvuWGtCw81PjbuU0gY5SfldXWbITDulelaLfs7a+g
JwIhAMXGpvdj/E7y5ADM5ZIkIZv7C9dgqu221Msn6NKa3puw
-----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:
                    04:16:71:8a:a4:36:ed:36:4d:c8:b6:97:b9:bb:82:
                    8f:b0:84:e4:5a:13:ec:e2:09:60:5e:eb:e6:0c:f3:
                    97:56:7b:81:f9:0a:d4:87:3d:cc:1e:b4:32:44:ce:
                    99:7a:9b:a3:b7:fb:0f:e5:36:02:20:03:22:16:75:
                    d1:23:d2:c3:15
                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
         30:45:02:20:10:be:e5:86:b4:2c:3c:d4:f8:db:b9:4d:20:63:
         94:9f:95:d5:d6:6c:84:c3:ba:57:a5:68:b7:ec:ed:af:a0:27:
         02:21:00:c5:c6:a6:f7:63:fc:4e:f2:e4:00:cc:e5:92:24:21:
         9b:fb:0b:d7:60:aa:ed:b6:d4:cb:27:e8:d2:9a:de:9b:b0
```

B.2.3. Key Pairs

B.2.3.1. Root Key Pair

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIFQg8T7V348htdP6kyA5luN62FEbsZUsp+mNfff68gK5oAoGCCqGSM49
AwEHoUQDQgAEFnGKpDbtNk3Itpe5u4KPsITkWhPs4glgXuvmDPOXVnuB+QrUhz3M
HrQyRM6Zepujt/sP5TYCIAMiFnXRI9LDFQ==
-----END EC PRIVATE KEY-----
Private-Key: (256 bit)
priv:
    54:20:f1:3e:d5:df:8f:21:b5:d3:fa:93:20:39:96:
    e3:7a:d8:51:1b:b1:95:2c:a7:e9:8d:7d:f7:fa:f2:
    02:b9
pub:
    04:16:71:8a:a4:36:ed:36:4d:c8:b6:97:b9:bb:82:
    8f:b0:84:e4:5a:13:ec:e2:09:60:5e:eb:e6:0c:f3:
    97:56:7b:81:f9:0a:d4:87:3d:cc:1e:b4:32:44:ce:
    99:7a:9b:a3:b7:fb:0f:e5:36:02:20:03:22:16:75:
    d1:23:d2:c3:15
ASN1 OID: prime256v1
NIST CURVE: P-256
```

B.2.3.2. Intermediate Key Pair

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEINVK+bWvkojgcgKWe9uZvgldgZKxv3iWfkN3tj5oXA+doAoGCCqGSM49
AwEHoUQDQgAEPxERrJqhwkOOZthBwU+Hat81DqpzbIlEX0IZs1z48zfInWTsZbEW
TcfmiWBiatXVrTch02A5OBsvAUtDCUyFsQ==
-----END EC PRIVATE KEY-----
Private-Key: (256 bit)
priv:
    00:d5:4a:f9:b5:af:92:88:e0:72:02:96:7b:db:99:
    be:09:5d:81:92:b1:bf:78:96:7e:43:77:b6:3e:68:
    5c:0f:9d
pub:
    04:3f:11:11:ac:9a:a1:c2:43:8e:66:d8:41:c1:4f:
    87:6a:df:35:0e:aa:73:6c:89:44:5f:42:19:b3:5c:
    f8:f3:37:c8:9d:64:ec:65:b1:16:4d:c7:e6:89:60:
    62:6a:d5:d5:ad:37:21:d3:60:39:38:1b:2f:01:4b:
    43:09:4c:85:b1
ASN1 OID: prime256v1
NIST CURVE: P-256
```

B.2.3.3. Leaf Key Pair

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIGvMiauzTwqDQUNpuQZB/B6nqRA+bAwwc8Yd5umOxEKtoAoGCCqGSM49
AwEHoUQDQgAEqbD4ZrAtkSuHZCJX0q4LB+H6g6aOtE8/FnlDo+XYAHLcCtWzAKT7
i8AFO018nY1Iu6xGjOUot1scWvouTdo4RQ==
-----END EC PRIVATE KEY-----
Private-Key: (256 bit)
priv:
    6b:cc:89:ab:b3:4f:0a:83:41:43:69:b9:06:41:fc:
    1e:a7:a9:10:3e:6c:0c:30:73:c6:1d:e6:e9:8e:c4:
    42:ad
pub:
    04:a9:b0:f8:66:b0:2d:91:2b:87:64:22:57:d2:ae:
    0b:07:e1:fa:83:a6:8e:b4:4f:3f:16:79:43:a3:e5:
    d8:00:72:dc:0a:d5:b3:00:a4:fb:8b:c0:05:3b:4d:
    7c:9d:8d:48:bb:ac:46:8c:e5:28:b7:5b:1c:5a:fa:
    2e:4d:da:38:45
ASN1 OID: prime256v1
NIST CURVE: P-256
```

B.3. Example Authentication Signature Verification

B.3.1. CHALLENGE Request

```
01 83 00 00      ; CHALLENGE Request Header
46 29 65 BE EE 5B 63 45 B6 F6 31 72 A2 53 5A 35    ; Nonce
A3 D5 73 A4 45 F6 E0 3F B9 DB AA 43 FE DD A0 AF
```

B.3.2. CHALLENGE_AUTH Response

```
01 03 00 01      ; CHALLENGE_AUTH Response Header
01 01 01 00      ; Min/MaxProtocolVersion, Capabilities
66 09 26 B6 CB 61 86 5C 60 78 1A 98 92 AB F4 B7    ; CertChainHash
C2 4A B6 27 7C 2A 69 84 8A C6 90 B4 1C 18 63 E1
C2 6C 8B 49 AC 8C 52 48 4F C0 43 14 51 E4 98 F5    ; Salt
2D E5 32 0C DA B7 A0 B7 92 38 E6 44 54 2B 49 AC
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ; Context Hash
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3C C4 AD FF 1B AC 89 14 4C F1 F5 6C AA 1A 96 CB    ; Signature
DD C9 EC FD D5 AA 4F 7C E5 E5 91 D1 89 14 BF EB
99 FA D2 96 5C 09 B3 71 36 2E EF 8D 1C 6D 9D 87
07 A6 63 C6 5D 9B 8C 2F 25 D8 BE B1 83 B8 6F 63
```

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 know 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.