Universal Serial Bus Security Foundation Specification

Revision 1.0 with ECN and Errata through January 7, 2019

CONTENTS

Specification Work Group Chairs / Specification Editors 5
Specification Work Group Contributors 5
Revision History 7
1 Introduction 8
1.1 Scope 8
1.2 Overview 8
1.3 Related Documents 9
1.4 Terms and Abbreviations 11
1.5 Conventions 12
1.5.1 Precedence 12
1.5.2 Keywords 12
1.5.3 Numbering 13
1.5.4 Byte Ordering 13
2 Overview 14
2.1 Cryptographic Methods 14
2.1.1 Random Numbers 14
2.2 Security Overview 14
2.2.1 Periodic Re-Authentication 14
2.2.2 Secret Key Storage and Protection 14
2.2.3 Security Evaluation Criteria 15
3 Authentication Architecture 16
3.1 Certificates 16
3.1.1 Format 16
3.1.2 Textual Format 16
3.1.3 Attributes and Extensions 16
3.2 Certificate Chains 16
3.2.1 Provisioning 17
3.3 Private Keys 17
4 Authentication Protocol 18
4.1 Digest Query 18
4.2 Certificate Chain Read 18
4.3 Authentication Challenge 18
4.4 Errors and Alerts 18
4.4.1 Invalid Request 18
4.4.2 Unsupported Protocol Version 19
4.4.3 Busy 19
4.4.4 Unspecified 19
5 Authentication Messages 20
5.1 Header 20
5.1.1 USB Security Foundation Protocol Version 20
5.1.2 Message Type 20

5.1.3 Param1 21
5.1.4 Param2 21
5.2 Authentication Requests 21
5.2.1 GET_DIGESTS 21
5.2.2 GET_CERTIFICATE 21
5.2.3 CHALLENGE 22
5.3 Authentication Responses 22
5.3.1 DIGESTS 23
5.3.2 CERTIFICATE 24
5.3.3 CHALLENGE_AUTH 24
5.3.4 ERROR 25
A Potential Attack Vectors 27

TABLES

Table 1-1: Terms and Abbreviations 11
Table 2-1: Summary of Cryptographic Methods 14
Table 3-1: Certificate Chain Format 16
Table 5-1: Authentication Message Header 20
Table 5-2: USB Security Foundation Protocol Version 20
Table 5-3: Authentication Request Types 21
Table 5-4: GET_DIGESTS Request Header 21
Table 5-5: GET_CERTIFICATE Request Header 22
Table 5-6: GET_CERTIFICATE Request Payload 22
Table 5-7: CHALLENGE Request Header 22
Table 5-8: CHALLENGE Request Payload 22
Table 5-9: Authentication Response Types 23
Table 5-10: DIGESTS Response Header 23
Table 5-11: DIGESTS Response Payload 23
Table 5-12: CERTIFICATE Response Header 24
Table 5-13: CERTIFICATE Response Payload 24
Table 5-14: CHALLENGE_AUTH Response Header 24
Table 5-15: CHALLENGE_AUTH Response Payload 25
Table 5-16: Message Contents for ECDSA Digital Signature 25
Table 5-17: ERROR Response Header 26
Table 5-18: ERROR Codes 26

1 Specification Work Group Chairs / Specification Editors

Renesas Electronics Corp.	Co-Chair	Bob Dunstan
Intel Corporation	Co-Chair	Abdul Ismail
	Editor	Stephanie Wallick

2 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		

1

2

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
1.0+ ECN and Errata	January 7, 2017	Includes ECN and errata through January 7, 2019 and separates authentication protocol and messages from USB-specific mappings.

1 Introduction

This specification provides a means for authenticating products with regard to identification and configuration. The methods for authentication can be used over multiple transports. It is intended that the mapping of the authentication methods to a specific transport be defined in a separate specification. See for example the USB Type-C Authentication Specification.

1.1 Scope

This specification defines the architecture and methodology for unilateral Authentication. Information is provided to allow for Policy enforcement, but individual Policy decisions are not specified.

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

- ***ASN.1*** - ISO-822-1-4;
  - ITU-T X.680 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.680-201508-I!!PDF-E&type=items);
  - ITU-T X.681 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.681-201508-I!!PDF-E&type=items);
  - ITU-T X.682 (Available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.682-201508-I!!PDF-E&type=items);
  - ITU-T X.683 (Available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.683-201508-I!!PDF-E&type=items.)
- ***DER*** - ISO-8825-1; ITU-T X.690 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.690-201508-I!!PDF-E&type=items.)
- ***X509v3*** - ISO-9594-8; ITU-T X.509 (available at: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-201210-I!!PDF-E&type=items.)
- ***Common Criteria***:
  - Common Criteria for Information Technology Security Evaluation (available at: https://www.commoncriteriaportal.org/cc/), which includes:
    - Part 1: Common Criteria for Information Technology Security Evaluation; Part 1: Introduction and general model; April 2017; Version 3.1 Revision 5; CCMB-2017-04-001 (https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf)
    - Part 2: Common Criteria for Information Technology Security Evaluation; Part 2: Security functional components; April 2017; Version 3.1 Revision 5; CCMB-2017-04-002 (https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf)
    - Part 3: Common Criteria for Information Technology Security Evaluation; Part 3: Security assurance components; April 2017; Version 3.1 Revision 5; CCMB-2017-04-003 (https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf)
    - CEM: Common Methodology for Information Technology Security Evaluation: Evaluation methodology; April 2017; Version 3.1 Revision 5; CCMB-2017-04-004 (https://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R5.pdf)
  - Addenda: CC and CEM addenda: Exact Conformance, Selection-Based SFRs, Optional SFRs; May 2017; Version 0.5 (Draft, Initial release, for trial use); CCDB-2017-05-xxx (https://www.commoncriteriaportal.org/files/ccfiles/CCDB-2017-05-17-CCaddenda-Exact_Conformance.pdf)
- ***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)
- ***JIL/JHAS:***
    - "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)
- ***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: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf)

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

2 ### 1.4 Terms and Abbreviations

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

4 **Table 1-1: Terms and Abbreviations**

Term/Abbreviation	Definition
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.
Mapping Organization	A standards organization that defines a mapping of the USB Security Foundation Authentication Protocol to a specific transport. For example, USB-IF is the Mapping Organization for authentication over the USB PD, USB 2.0, and USB 3.2 transports (see the USB Type-C Authentication Specification).
Nonce	A number used only once in any given key context. Can be interpreted as N-Once.
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.
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.
Slot	The position of a Certificate Chain in an Authentication Responder. Each Slot corresponds to one Certificate Chain.

1.5 Conventions

1.5.1 Precedence

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

1.5.2 Keywords

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

1.5.2.1 Conditional Normative

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

1.5.2.2 Deprecated

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

1.5.2.3 Informative

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

1.5.2.4 May

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

1.5.2.5 N/A

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

1.5.2.6 Normative

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

1.5.2.7 Optional/Optionally/Optional Normative

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

1.5.2.8 Reserved

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

1.5.2.9 Shall/Normative

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

1.5.2.10 Should

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

1.5.3 Numbering

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

1.5.4 Byte Ordering

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

2 Overview

This section contains no Normative requirements.

2.1 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

2.1.1 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.

2.2 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 USB PD Sink may choose not to use the full advertised capabilities of an unauthenticated USB PD Source.

2.2.1 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.

2.2.2 Secret Key Storage and Protection

One threat concerning the USB Security Foundation authentication protocol 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 Security Foundation ecosystem.

Therefore, it is recommended that vendors take appropriate measures to protect the execution of the USB Security Foundation 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 A.

2.2.3 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.

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.

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

Certificate attributes and extensions are defined by the Mapping Organization. See for example the USB Type-C Authentication Specification.

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.

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.

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 an Authentication Responder acquires one or more Certificate Chains. The procedure for provisioning an Authentication Responder is outside the scope of this specification.

3.3 Private Keys

Each Certificate Chain in an Authentication Responder corresponds to a private key whose corresponding public key is certified in the Leaf Certificate of that Slot. An Authentication Responder must have access to that private key. All Private Keys shall be unique and shall be generated, provisioned, and stored in a manner that adequately protects the confidentiality of the key.

A private key used by one Authentication Responder shall not be used by any other Authentication Responders. For example, one instance of a USB PD power supply cannot have the same private key as another instance of the USB PD power supply, even if they are otherwise identical model

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 an Authentication Responder, 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 the timing requirements, 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.

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 Security Foundation Protocol Version

This field identifies which version of the USB Security Foundation Specification is being used. Bits 0 through 3 (lower nibble) identify the Minor Revision number and bits 4 through 7 (upper nibble) identify the Major Revision Number.

Table 5-2 shows the valid values for this field. A product shall not use a USB Security Foundation Protocol Version value corresponding to a specification revision that it does not support.

Table 5-2: USB Security Foundation Protocol Version

Name	Value	Meaning
Reserved	00–0Fh	Reserved
V1.0*	10h	USB Security Foundation Authentication Protocol Version 1.0
Reserved	20h–FFh	Reserved

*a Value of 01h also indicates V1.0

It is intended in the future that products support a contiguous range of USB Security Foundation 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 (n^{th}) 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	Certificate Chain segment of requested slot number and starting at requested offset. Segment length shall be less than or equal to requested length, and greater than or equal to 1.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	*OrgName*	1	Identifies the Mapping Organization using the following encodings: 0 USB-IF 1 - 15 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	32-byte ***SHA256*** hash of product-specific information. The product-specific information is defined by the Mapping Organization. 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*.

A 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.