Huawei BaseBIOS Security Target

Huawei Technologies Co., Ltd

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base

Bantian, Longgang

Shenzhen 518129

People’s Republic of China

Website: http://www.huawei.com

Change History

Date	Version	Description
2020-04-15	1.0	Initial version
2020-06-16	1.1	Change version number
2020-07-09	1.2	Clarify the intended device type and correct table numbers
2020-07-15	1.3	Removal of FCS_COP.1 instances. Correction on CC claims. Editorial changes.
2020-07-28	1.4	Add descriptions about debug function
2020-09-04	1.5	Correction on version references, delivery format, CC claims
2020-10-06	1.6	Correction on required non-TOE parts, logical scope

Content

List of tables

Abbreviations

Abbreviation	Full Name	Description
BSBC	Boot-rom Secure Boot Code	The chiprom boot code.
ST	Security Target	
TOE	Target Of Evaluation	
SP	Secure Partition	Secure Partition
eFUSE	Electrically Programmable Fuse	
SEC	Secure Engine for Symmetric Algorithm	Hardware accelerator for AES Algorithm
CER	Certification Engine for asymmetric algorithm	Hardware accelerator for SHA Algorithm and RSA Algorithm
KM	Key management	
AES	Advanced Encryption Standard	The symetric key cryptography algorithm
RSA	Rivest Shamir Adleman	The public key cryptography algorithm invented by Rivest、Shamir and Adleman.

1 Introduction

This introduction chapter contains the following sections:

1.1 ST Reference and TOE Reference

1.2 TOE Overview

1.3 TOE Description

1.1 ST Reference and TOE Reference

1.1.1 ST Reference

Title:	Huawei BaseBIOS Security Target
Author:	Huawei Hardware Trusted Technology and Engineering Lab
Version:	1.6
CC Version:	Version 3.1, Revision 5
Assurance Level:	EAL4+
ITSEF:	Brightsight
Certification Body:	Netherlands Scheme for Certification in the Area of IT Security (NSCIB)

1.1.2 TOE Reference

Name:	BaseBIOS
Developers:	Huawei Hardware Trusted Technology and Engineering Lab
Version:	1.5

1.2 TOE Overview

1.2.1 TOE Type

The TOE is BaseBIOS which is a secure bootloader, it is the second piece of software codes that is used in the start-up process of a secure embedded hardware product, such as an integrated secure element or SoC, to ensure the product securely initializes into a secure state.

1.2.2 TOE Usage and Major Security Features

The TOE is used by integrating it into a software stack running on a compatible hardware platform. As part of the initialization procedure of this hardware and software stack, the TOE will be used to ensure that the higher layers of software are securely loaded.

In order to support this, the TOE provides the following security features:

- Boot failure handling
- Sensitive data handling
- Debug functionality (for software development)

1.2.3 Required non-TOE hardware/software/firmware

The TOE requires a hardware platform including firmware that provides the following functionality:

- One-time programmable memory
- Cryptographic operations for software image decryption and signature verification

Additionally, the TOE requires a target software image for loading, encrypted and signed using the appropriate cryptographic schemes.

The TOE is located in Secure Partition (SP), which ensures the different layers of software are securely loaded. The TOE is only a piece of secure bootloader code and stored in SRAM of SP. Initially, the encrypted TOE is stored in the Flash, which is an external component that is not part of the same hardware that contains the SP. The first stage bootloader, which is stored in the ROM of SP, decrypts and authenticates the TOE and stores it in SRAM, where it will run. The TOE also relies on the External cache, which is not part of the SP, but is still part of the same hardware that contains the SP (whereas the Flash is not). Figure 1 shows the TOE in its operational environment. The following summarises the role of the various components:

- External Cache and SRAM are used by the TOE as volatile memory.
- KM and SEC are used for image decryption.
- CER is used for image authentication (signature verification).
- eFUSE is used as one-time programmable memory.
- CPU executing the TOE.

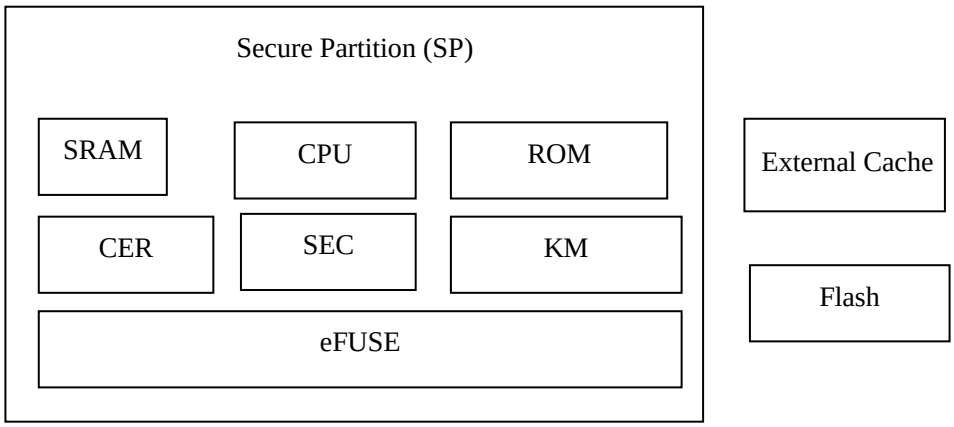

Figure 1 TOE and Operational Environment

1.3 TOE Description

1.3.1 Physical Scope of TOE

The TOE consists of the following components:

Table 1 Components of the TOE scope

Type	Component	Version	Format	Delivery method
Software	Huawei BaseBIOS	V1.5	Binary(.bin)	Encrypted file exchange
Guidance document	Huawei BaseBIOS CC EAL4+ Guidance Document	V1.3	Document (.pdf)	Encrypted file exchange

1.3.2 Logical Scope of TOE

The logical scope of the TOE consists of the following security features:

- SF.Boot,

  which implements the boot process, including:

  - the handling of hardware initialization failure and software authentication failure, and
  - the clearing of sensitive data, for both a successful and unsuccessful boot.

- SF.Debug
  - which implements mechanisms to disable the debug functionality after software development is completed.

1.3.3 TOE Life Cycle

The life cycle of the TOE is divided into the following three phases:

- Phase 1 corresponds to the design of the TOE
- Phase 2 corresponds to the develop and the test of the TOE
- Phase 3 corresponds to the delivery of the TOE

Table 2 shows the whole life cycle of the TOE.

Table 2 Actors in the TOE Life Cycle

Phase	Actors
1：the design of the TOE	The TOE developer：design the architecture of the TOE
2：the develop and the test of the TOE	The TOE developer: develop and test of the TOE

3：the delivery of the TOE	The TOE developer: write the usage guidelines documents，encrypt and package the TOE

2 Conformance Claims

This chapter is divided into the following sections:

2.1 CC Conformance Claim

2.2 PP Conformance Claim

2.3 Conformance Claim Rationale

2.1 CC Conformance Claim

This Security Target conforms to CC Part 2 extended and Part 3 conformant, with a claimed Evaluation Assurance Level of EAL 4, augmented by ALC_DVS.2 and AVA_VAN.5.

This Security Target claims conformance to the following specifications：

- Common Criteria for Information Technology Security Evaluation, Part 2: Security Function Requirements; Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements; Version 3.1, Revision 5, April 2017

2.2 PP Conformance Claim

This Security Target does not claim conformance to any PP.

2.3 Conformance Claim Rationale

This is not applicable as this Security Target, because it does not claim conformance to any PP.

3 Security Problem Definition

This Security Target claims conformance to the CC version 3.1 revision 5. Assets, threats, assumptions and organizational security policies are taken from CC. This chapter lists these assets, threats, assumptions and organizational security policies, and describes extensions to these elements in detail.

3.1 Assets

The assets of the TOE are divided into the following:

- The user software to be loaded, which is stored in external NVM.
- User cryptographic keys, which are stored in OTP or derived from such keys.

3.2 Threats

The threats of the TOE are divided into the following:

T.Software-Confidentiality

An unauthorized external entity discloses the user software to be loaded through logical means, including but not limited to abuse of debug functionality.

T.Software-Integrity

An unauthorized external entity loads unauthentic software through logical means, including but not limited to changing the user software while stored in external NVM and/or abuse of debug functionality.

T.Key-Confidentiality

An unauthorized external entity discloses the user cryptographic keys (or keys that are derived from such keys) through logical means, including but not limited to abuse of debug functionality.

3.3 Assumptions

This section states the assumptions that hold on the TOE operational environment. These assumptions have to be satisfied.

A.Environment-Protection

It is assumed that the TOE is integrated into a hardware environment that protects against the following attacks:

- Physical attacks, such as probing or physical manipulation.
- Malfunction attacks, where environmental stress is applied to cause a malfunction.
- Side-channel attacks, where sensitive information leaks as a result of leakage.

A.Hardware-Support

It is assumed that the TOE is integrated into a hardware environment that provides (at a minimum) the

following functionality:

- One-Time Programmable (OTP) memory.
- Cryptographic functionality, which includes
    - RSASSA-PKCS1-v1_5 and RSASSA-PSS as defined by [PKCS#1] with key sizes 2048 and 4096 bits,
    - AES-GCM as defined by [SP800-38D] with key size 256 bits,
    - SHA-256 as defined by [FIPS180-4],
    - HMAC-SHA-256 as defined by [FIPS198-1], and
    - PBKDF2 as defined by [SP800-132].

A.Process-Security

It is assumed that security procedures are used after delivery of the TOE by the TOE Manufacturer up to delivery to the end-consumer to maintain confidentiality and integrity of the TOE and of its manufacturing and test data (to prevent any possible copy, modification, retention, theft or unauthorised use).

A.Process-User-Data

All data that is considered user data from the point of view of this TOE needs to be treated according to its security needs by the user when handled outside of the TOE. Specifically, the user software to be loaded is assumed to be stored in the external NVM in a way that preserves its confidentiality and supports the authentication of the user software by TOE. Additionally, the user keys are assumed to be handled securely when outside of the control of the TOE.

4 Security Objectives

This chapter contains the following sections:

4.1 Security Objectives for the TOE

4.2 Security Objectives for the Operational Environment

4.3 Security Objective Rationale

4.1 Security Objectives for the TOE

The TOE shall provide the following security objectives:

O.Authentication

The TOE shall authenticate the user software (with the support of the hardware platform as described in the following section). In case any failure occurs during the authentication of the loaded user data, the TOE shall preserve a secure state.

O.Data

The TOE shall restrict the access to sensitive data and keys by clearing it from memory, both when the user data is loaded correctly and when a failure occurs.

O.Abuse-Debug

The TOE shall implement functionality for a user to limit the use of debug functionality, to prevent its use to disclose or manipulate sensitive data in the operational environment of the Composite TOE.

4.2 Security Objectives for the Operational Environment

The following security objectives for the operational environment are specified according to the CC.

OE.Environmental-Protection

The hardware environment that the TOE is integrated into shall protect against the following attacks:

- Physical attacks, such as probing or physical manipulation
- Malfunctions, caused by deliberate environmental stress applied by an attacker
- Side-channel attacks, where sensitive information leaks through operational parameters

OE.Hardware-Support

The hardware environment that the TOE is integrated into shall provide (at a minimum) the following functionality:

- One-Time Programmable (OTP) memory.
- Cryptographic functionality, which includes
    - RSASSA-PKCS1-v1_5 and RSASSA-PSS as defined by [PKCS#1] with key sizes 2048 and

4096 bits,

- AES-GCM as defined by [SP800-38D] with key size 256 bits,
- SHA-256 as defined by [FIPS180-4],
- HMAC-SHA-256 as defined by [FIPS198-1], and
- PBKDF2 as defined by [SP800-132].

OE.Process-Security

Security procedures shall be used after TOE Delivery up to delivery to the end-consumer to maintain confidentiality and integrity of the TOE and of its manufacturing and test data (to prevent any possible copy, modification, retention, theft or unauthorised use).

OE.Process-User-Data

All data that is considered user data from the point of view of this TOE shall be treated according to its security needs by the user when handled outside of the TOE. Specifically, the user software to be loaded shall be stored in the external NVM encrypted using AES-256-GCM (as defined by [SP800-38D]) and shall be stored together with a digital signature according to RSASSA-PSS or RSA-PKCS1-v1_5 (as defined by [PKCS#1]) with key size 2048 or 4096 bits and hash function SHA-256 (as defined by [FIPS180-4]). Additionally, the user keys shall be handled securely when outside of the control of the TOE.

4.3 Security Objective Rationale

4.3.1 Overview

Table 3 shows an overview of the tracing from objectives to Threats and Assumptions. In the subsequent sections this tracing is explained in more detail.

Table 3 Tracing from Security Objectives to Threats and Assumption

Security Objectives / Security Problem Definition	O.Authentication	O.Data	O.Abuse-Debug	OE.Environmental-Protection	OE.Hardware-Support	OE.Process-Security	OE.Process-User-Data
T.Software-Confidentiality	√	√	√		√		
T.Software-Integrity	√		√		√		
T.Key-Confidentiality		√	√		√		
A.Environmental-Protection				√			
A.Hardware-Support					√		
A.Process-Security						√	
A.Process-User-Data							√

4.3.2 Threats

The threat **T.Software-Confidentiality** is countered by the authentication process (O.Authentication), which is supported by cryptographic functionality from the hardware (OE.Hardware-Support).

Specifically, this process includes decryption of the user software that is stored in encrypted form in external NVM. Additionally, the TOE clears all sensitive data and keys that have been stored in memory during the authentication process before this memory is released to other processes (O.Data). Finally, the TOE prevents an attacker from using debug functionality to disclose such data (O.Abuse-Debug), relying on the OTP memory provided by the hardware (OE.Hardware-Support).

The threat **T.Software-Integrity** is countered by the authentication process (O.Authentication), which is supported by cryptographic functionality from the hardware (OE.Hardware-Support). Specifically, this process includes verification of the digital signature that is stored together with the user software in external NVM. Additionally, the TOE prevents an attacker from using debug functionality to manipulate such data (O.Abuse-Debug), relying on the OTP memory provided by the hardware (OE.Hardware-Support).

The threat **T.Key-Confidentiality** is countered by the fact that the TOE clears all sensitive data and keys that have been stored in memory during the authentication process before this memory is released to other processes (O.Data). Additionally, the TOE prevents an attacker from using debug functionality to disclose such data (O.Abuse-Debug), relying on the OTP memory provided by the hardware (OE.Hardware-Support).

4.3.3 Assumptions

The assumption **A.Environmental-Protection** is directly met by its corresponding objective for the environment **OE.Environmental-Protection**.

The assumption **A.Hardware-Support** is directly met by its corresponding objective for the environment **OE.Hardware-Support**.

The assumption **A.Process-Security** is directly met by its corresponding objective for the environment **OE.Process-Security**.

The assumption **A.Process-User-Data** is directly met by its corresponding objective for the environment **OE.Process-User-Data**.

5 Extended Components Definition

5.1 Definition of the Family FMT_LIM

To define the IT security functional requirements of the TOE an additional family (FMT_LIM) of the Class FMT (Security Management) is defined here, identical to its definition in PP0084. This family describes the functional requirements for the Test Features of the TOE. The new functional requirements were defined in the class FMT because this class addresses the management of functions of the TSF. The examples of the technical mechanism used in the TOE appropriate to address the specific issues of preventing the abuse of functions by limiting the capabilities of the functions and by limiting their availability.

The family “Limited capabilities and availability (FMT_LIM)” is specified as follows.

FMT_LIM Limited capabilities and availability

Family behaviour

This family defines requirements that limit the capabilities and availability of functions in a combined manner. Note that FDP_ACF restricts the access to functions whereas the component Limited Capability of this family requires the functions themselves to be designed in a specific manner.

Component levelling:

FMT_LIM.1 Limited capabilities requires that the TSF is built to provide only the capabilities (perform action, gather information) necessary for its genuine purpose.

FMT_LIM.2 Limited availability requires that the TSF restrict the use of functions (refer to Limited capabilities (FMT_LIM.1)). This can be achieved, for instance, by removing or by disabling functions in a specific phase of the TOE’s life-cycle.

Management: FMT_LIM.1, FMT_LIM.2

There are no management activities foreseen.

Audit: FMT_LIM.1, FMT_LIM.2

There are no actions defined to be auditable.

The TOE Functional Requirement “Limited capabilities (FMT_LIM.1)” is specified as follows.

FMT_LIM.1 Limited capabilities

Hierarchical to: No other components.

Dependencies: FMT_LIM.2 Limited availability.

FMT_LIM.1.1 The TSF shall be designed and implemented in a manner that limits its capabilities so that in conjunction with “Limited availability (FMT_LIM.2)” the following policy is enforced [assignment: Limited capability policy].

The TOE Functional Requirement “Limited availability (FMT_LIM.2)” is specified as follows.

FMT_LIM.2 Limited availability

Hierarchical to: No other components.

Dependencies: FMT_LIM.1 Limited capabilities.

FMT_LIM.2.1 The TSF shall be designed in a manner that limits its availability so that in conjunction with “Limited capabilities (FMT_LIM.1)” the following policy is enforced [assignment: Limited availability policy].

Application Note: The functional requirements FMT_LIM.1 and FMT_LIM.2 assume that there are two types of mechanisms (limitation of capabilities and limitation of availability) which together shall provide protection in order to enforce the same policy or two mutual supportive policies related to the same functionality. This allows e.g. that

(i) The TSF is provided without restrictions in the product in its user environment but its capabilities are so limited that the policy is enforced or conversely

(ii) The TSF is designed with high functionality but is removed or disabled in the product in its user environment.

6 Security Requirements

This part of the Security Target defines the detailed security requirements that shall be satisfied by the TOE. The statement of TOE security requirements shall define the functional and assurance security requirements that the TOE needs to satisfy in order to meet the security objectives for the TOE. This chapter contains the following sections:

6.1 Statement of Security Requirements

6.2 Security Requirements Rationale

6.1 Statement of Security Requirements

6.1.1 Notation

The CC allows several operations to be performed on security requirements (on the component level). Refinement, selection, assignment and iteration are defined in paragraph 8.1 of Part 1 of the CC.

The assignment operation and the selection operation are used in this Security Target, Neither the iteration operation nor the refinement operation has been performed.

The assignment operation is used to assign a specific value to an unspecified parameter, such as the length of a password. Assignments having been made are denoted by showing as italic text.

The selection operation is used to select one or more options provided by the CC in stating a requirement. Selections having been made are denoted as italic text.

6.1.2 Security Functional Requirements

The security functional requirements are as follows:

FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

- *Failure to initialise the hardware*
- *Failure to authenticate the loaded software*
- *An exception occurs*

FDP_RIP.1 Subset residual information protection

FDP_RIP.1.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the *deallocation of the resource from* the following objects: *all keys and sensitive data stored in SRAM.*

FMT_LIM.1 Limited capabilities

FMT_LIM.1.1 The TSF shall be designed and implemented in a manner that limits its capabilities so that in conjunction with “Limited availability (FMT_LIM.2)” the following policy is enforced *Deploying Debug Features after Composite TOE Delivery does not allow*

user data of the Composite TOE to be disclosed or manipulated, TSF data to be disclosed or manipulated, software to be reconstructed and no substantial information about construction of TSF to be gathered which may enable other attacks.

FMT_LIM.2 Limited availability

FMT_LIM.2.1 The TSF shall be designed in a manner that limits its availability so that in conjunction with "Limited capabilities (FMT_LIM.1)" the following policy is enforced *Deploying Debug Features after Composite TOE Delivery does not allow user data of the Composite TOE to be disclosed or manipulated, TSF data to be disclosed or manipulated, software to be reconstructed and no substantial information about construction of TSF to be gathered which may enable other attacks.*

6.1.3 Security Assurance Requirements

The claimed Security Assurance Requirements are given by the package claim in Section 2.1, and the corresponding SARs are included here by reference to Part 3.

6.2 Security Requirements Rationale

6.2.1 Security Functional Requirements

6.2.1.1 Dependencies

Table 4 shows the dependencies of the Security Functional requirements, whether they are met, and, if not, the justification.

Table 4 Security Functional Requirements dependencies

SFR	Dependencies	Met by	Justification
FPT_FLS.1	-	-	N/A
FDP_RIP.1	-	-	N/A
FMT_LIM.1	FMT_LIM.2	FMT_LIM.2	N/A
FMT_LIM.2	FMT_LIM.1	FMT_LIM.1	N/A

6.2.1.2 Tracing

Table 5 shows an overview of the tracing from SFRs to the Security Objectives. The details are provided in the following.

Table 5 Tracing from Security Functional Requirements to Security Objectives

Security Functional Requirements / Security Objectives	FPT_FLS.1	FDP_RIP.1	FMT_LIM.1	FMT_LIM.2
O.Authentication	√			
O.Data	√	√		
O.Abuse-Debug			√	√

The Security Objective **O.Authentication** is that the TOE shall authenticate the user software with the support of the hardware. To this end, it needs to preserve a secure state in case of failures, which is directly met by the requirement FPT_FLS.1.

The Security Objective **O.Data** is that the TOE shall clear sensitive data from memory when it is no longer used, which is directly met by the requirement FDP_RIP.1, and in case of a failure, which is directly met by the requirement FPT_FLS.1.

The Security Objective **O.Abuse-Debug** is that the TOE shall implement a way for a user to limit the debug functionality such that it cannot be used by an attacker in the operational environment of the TOE to disclose or manipulate sensitive data. This is directly met by the combination of the requirements that this debug functionality is either not useful to an attacker, as per FMT_LIM.1, or is made unavailable to an attacker, as per FMT_LIM.2.

6.2.2 Security Assurance Requirements

6.2.2.1 Dependencies

The dependencies of the claimed package are internally consistent. Additionally, Table 6 shows that the dependencies for the augmentations are met as well.

Table 6 Security Assurance Requirements dependencies for augmentations

SFR	Dependencies	Met by	Justification
AVA_VAN.5	ADV_ARC.1	ADV_ARC.1	Part of EAL4
	ADV_FSP.4	ADV_FSP.4	Part of EAL4
	ADV_TDS.3	ADV_TDS.3	Part of EAL4
	ADV_IMP.1	ADV_IMP.1	Part of EAL4

	AGD_OPE.1	AGD_OPE.1	Part of EAL4
	AGD_PRE.1	AGD_PRE.1	Part of EAL4
ALC_DVS.2	-	-	N/A

6.2.2.2 Justification

An assurance level of EAL4 with the augmentations AVA_VAN.5 and ALC_DVS.2 are required for this TOE since it is intended to defend against sophisticated attacks. Maximum assurance can be gained from positive security engineering based on good commercial practices. In order to provide a meaningful level of assurance that the TOE provides an adequate level of defence against such attacks, the evaluators will have access to the low level design and source code.

7 TOE summary specification

This chapter provides information to potential users of the TOE how the TOE satisfies the Security Functional Requirements.

7.1 SF.Boot

SF.Boot implements the boot process, including the handling of hardware initialization failure, software authentication failure, and any other unrecoverable exception. In case such a failure occurs, SF.Boot ensures that a secure state is maintained by

- Aborting the boot process
- Clearing any sensitive data

In case such failures do not occur, SF.Boot nevertheless clears any sensitive data, and finally cedes control to the authenticated user software. Therefore, SF.Boot ensures that the TOE meets FPT_FLS.1 and FDP_RIP.1.

7.2 SF.Debug

SF.Debug, which implements mechanisms to completely disable the debug functionality after software development is completed. These mechanisms ensure that an exception is generated in case a debug access is attempted after the debug functionality has been disabled, resulting in a boot failure. This means that an attacker can use the debug functionality to compromise user assets, and as such SF.Debug ensures that the TOE meets FMT_LIM.1 and FMT_LIM.2.

8 Bibliography

8.1 Evaluation Documents

[1] Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, Version 3.1, Revision 5, April 2017

[2] Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017

[3] Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

[4] Common Methodology for Information Technology Security Evaluation: Evaluation Methodology, Version 3.1, Revision 5, April 2017

8.2 Developer Documents

[5] Huawei BaseBIOS CC EAL4+ Guidance Document

8.3 Other Documents

[6] [FIPS180-4], FIPS Publication 180-4, Secure Hash Standard (SHS), Federal Information Processing Standards, August 2015

[7] [FIPS198-1], FIPS Publication 198-1, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards, July 2008

[8] [PKCS#1], PKCS #1 v2.2: RSA Cryptography Standard, RSA Laboratories, October 27, 2012

[9] [SP800-38D], NIST Special Publication 800-38D, Recommendations for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, National Institute of Standards and Technology, November 2007

[10] [SP800-132], NIST Special Publication 800-132, Recommendations for Password-Based Key Derivation: Part 1: Storage Applications, December 2010