VaultIP

FIPS 140-2 Non-Proprietary Security Policy

Document Revision 1.2

Document Date: 2014-10-14

Prepared by:

atsec information security Corp.

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

INSIDE Secure

Arteparc Bachasson, Bât A

Rue de la carrière de Bachasson, CS70025

Meyreuil, Bouches-du-Rhône 13590

France

Phone: +31-73-6581900

Fax: +31-73-6581999

http://www.insidesecure.com/

For further information contact: ESSEmbeddedHW-Support@insidesecure.com

Copyrights and Trademarks

Table of contents

1 Introduction

This document is the non-proprietary FIPS 140-2 Security Policy for the **VaultIP** cryptographic module. It contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 2 module.

1.1 Purpose of the Security Policy

There are three major reasons that a security policy is needed:

- it is required for FIPS 140-2 validation,
- it allows individuals and organizations to determine whether a cryptographic module, as implemented, satisfies the stated security policy, and
- it describes the capabilities, protection and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

1.2 Target Audience

This document is part of the package of documents that are submitted for FIPS 140-2 conformance validation of the module. It is intended for the following audience:

- Developers.
- FIPS 140-2 testing lab.
- The Cryptographic Module Validation Program (CMVP).
- Designers of Application-Specific Integrated Circuits (ASIC) integrating the crypto module for end-user products like mobile communications and consumer electronics appliances.
- Customers using or considering integration of VaultIP.

2 Cryptographic Module Specification

2.1 Module Overview

VaultIP is a Silicon IP Security Module which includes a complete set of high-level and low-level cryptographic functions. It offers key management and crypto functions needed for platform and application security such as Content Protection and Mobile Payment, and can be used stand-alone or as a 'Root of Trust' to support a TEE-based platform.

VaultIP completely shields all key and security sensitive data from all CPUs, interfaces and memory. Security sensitive materials are stored as assets that never leave VaultIP in unencrypted and/or non-authenticated form.

Additionally, VaultIP offers hardware security features that are needed when operating in a Trusted Execution Environment (TEE). These features include One-Time-Programmable memory (OTP) access and management, Random Number Generation / entropy source, timers, (short) monotonic/non-volatile counters and import and export of keys and other assets.

The VaultIP module provides a slave and a master interface. The slave interface is used to receive commands from one or more host CPUs. The master interface is used for autonomous data reads and writes from and to an external memory, flash or interface.

VaultIP supports many FIPS-Approved or FIPS-Allowed cryptographic algorithms as shown in Table 11.

2.2 Intended Usage

VaultIP is primarily aimed to be integrated in the design of Application-Specific Integrated Circuits (ASIC). However, it can also be synthesized in a Field-Programmable Gate Array (FPGA). For the purpose of this Cryptographic Module Validation, VaultIP is synthesized in a Zynq-7000 All Programmable SoC (Z-7010, Z-7015, and Z-7020).

The primary application of VaultIP is in mobile communications and consumer electronics appliances, where authentication, encrypted content processing using standard protocols, and protection of keys and other sensitive assets are required. VaultIP is best suited for mobile phones, tablets, wireless handsets, PDA-like devices and set top boxes that have the resources and connectivity to download, store and play back digital media content. These small, battery-powered devices require a low power IP solution with these features available in VaultIP:

- Low-power and small footprint IP.
- Internal storage for protection and management of sensitive keys and assets.
- Root of Trust as true hardware interface to on chip One-Time Programmable (OTP) memory.
- Secure Timers (hardware counters).
- Encryption engines to offload computationally intensive symmetric algorithms: AES, Triple-DES.
- Hash engine to offload computational intensive hash algorithms: SHA-1, SHA-2.
- Public Key Encryption, supporting RSA, ECDSA (sub-)functions.
- True random number generator (TRNG), also known as Non-deterministic random number generator (NDRNG).

- Embedded DMA controller for high speed symmetric crypto and hash data transfer.

2.3 FIPS 140-2 Module Information

For the purpose of this Cryptographic Module Validation, VaultIP synthesized on the FPGA chip (Zynq-7000 All Programmable SoC) is defined as a single-chip hardware module validated at security level 2.

The table below shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2:

FIPS 140-2 Sections		Security Level
1	Cryptographic Module Specification	2
2	Cryptographic Module Ports and Interfaces	2
3	Roles, Services and Authentication	2
4	Finite State Model	2
5	Physical Security	3
6	Operational Environment	N/A
7	Cryptographic Key Management	2
8	EMI/EMC	2
9	Self Tests	2
10	Design Assurance	2
11	Mitigation of Other Attacks	N/A

Table 1: Security Levels

VaultIP has been tested on a Xilinx ZC702 base board including the XC7Z020-CLG484-1 FPGA (Zynq-7000 all Programmable SoC series)

2.4 Approved Modes of Operation

VaultIP has two modes of operation: FIPS mode and non-FIPS mode. Although all services are available in both modes of operation, VaultIP verifies that only permitted cryptographic algorithms, modes and key lengths are requested in each mode of operation:.

- In FIPS mode of operation, only FIPS-Approved or FIPS-Allowed cryptographic algorithms with specific modes and key lengths can be requested. Table 11 shows all algorithms supported by the module in FIPS mode.
- In non-FIPS mode of operation, all cryptographic algorithms can be requested. Table 12 shows the additional algorithms supported by the module in non-FIPS mode.

VaultIP automatically turns to non-FIPS mode after the power-up tests succeed. The user must send a *Self-Test* token to turn the module to FIPS mode; to turn to non-FIPS mode the user must reset the module using the *Reset* token.

VaultIP enforces the separation of keys and CSPs between FIPS mode and non-FIPS mode. The mode switching via the execution of Self-Test token or Reset token results in the zeroization of the asset store.

2.5 System Block Diagram

Figure 1 provides a system diagram in which VaultIP is integrated in a SoC (System on a Chip) with one or more CPUs and connects to a common bus system.

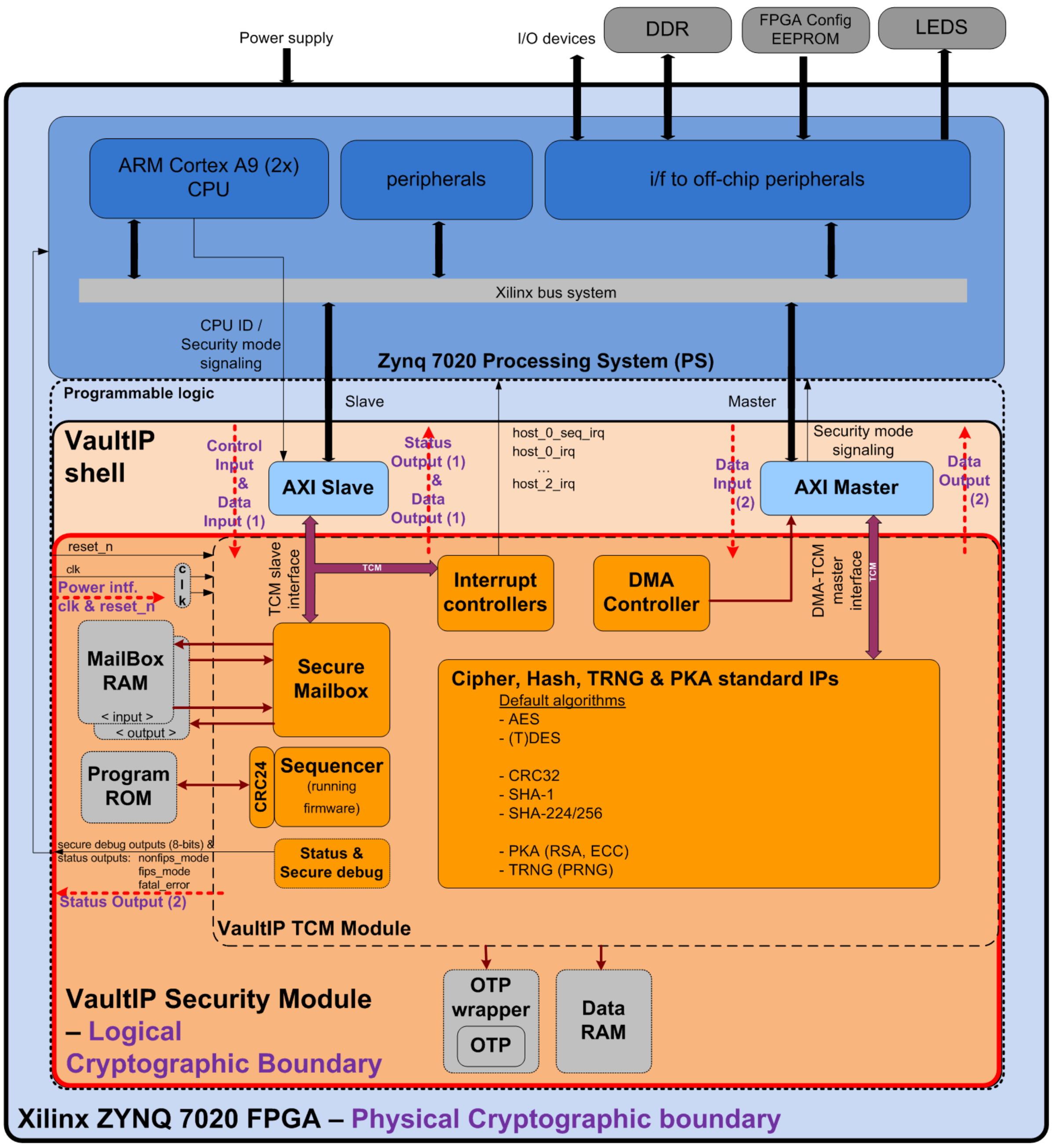

Figure 1 - System Block Diagram

For the purpose of this validation, the physical cryptographic boundary is enclosed in the FPGA. The logical cryptographic module boundary is represented with the red line box. The orange boxes represent the VaultIP components that comprise the IP core (the VaultIP firmware is stored in Program ROM). The grey boxes represent components that are provided in the IP core but must be replaced or adjusted during the synthesis process as they are technology dependent (OTP, RAM, ROM, etc.).

2.6 Hardware Block Diagram

VaultIP consists of two major components, the Verilog RTL and the Firmware running from a ROM on the sequencer. The RTL implements the cryptographic algorithms and basic public key long number mathematics. The Firmware handles the higher level operations, manages the keys and takes care of the data transfers by setting up DMAs.

The breakdown of VaultIP is shown in Figure 2, it shows the details of all interface that cross the security boundary and the first hierarchy levels of the VaultIP RTL.

Firmware is located in the Program ROM. The firmware has many routines; typically the sources for each routine are located in an individual assembly file. All Firmware routines are located on the same hierarchical level.

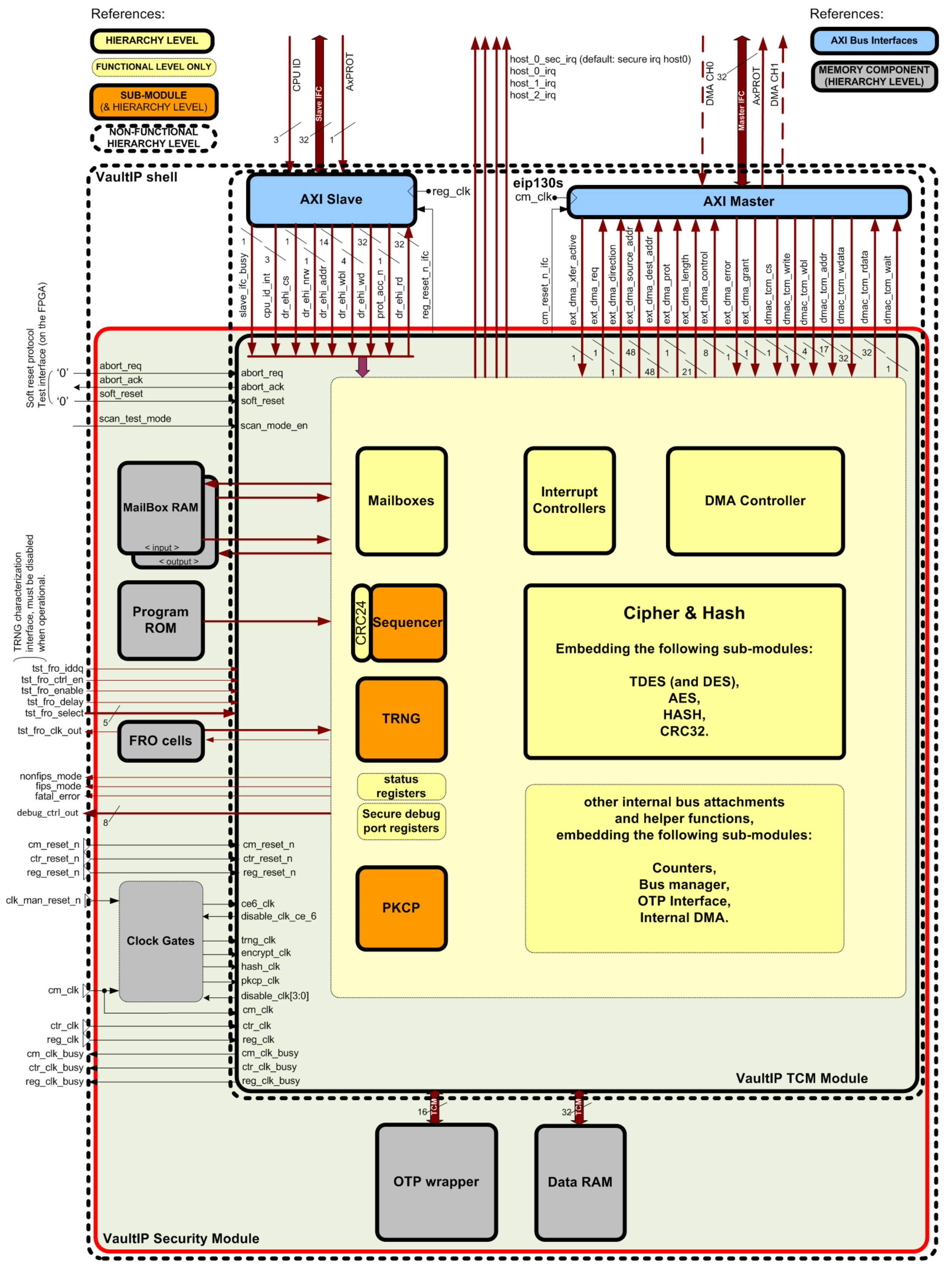

Figure 2 - Hardware Block Diagram

2.7 VaultIP module breakdown

The next list shows all (sub-)module levels of VaultIP and their corresponding version numbers. The levels provide an overview starting at the VaultIP shell level.

- Top-level VaultIP and sub-modules HW1.1.4.
  - VaultIP TCM Module
    Implementation of VaultIP TCM level. The TCM level embeds many sub-modules. Several components are not implemented as dedicated sub-modules from a design perspective, but do perform a specific operation and have their own hierarchy level:
    - CRC32
    - CRC24
    - Counters
    - Mailboxes
    - OTP Interface
    - Bus manager
    - Internal DMA

    The individual sub-modules from a hardware design perspective are listed below.
    - TDES and DES with ECB and CBC.
    - AES with ECB, CBC, CTR, CMAC, CCM, XTS and f8.
      - AES data path (ECB)
    - PKCP (public key co-processor), available for the Internal Controller to offload computationally intensive Public Key operations, such as modular exponentiations and Elliptic Curve Cryptography operations.
    - HASH, SHA-1 and SHA-2, including SHA-224 and SHA-256.
      - Multi-input 32-bit wide adders.
    - TRNG (capturing the entropy).
    - Sequencer (‘tiny’ RISC processor), running the Firmware code
    - DMA controller, requesting data reads or writes to or from VaultIP .
    - Interrupt controller, captures the various interrupt sources and manages these to a single host interrupt. Multiple instantiations.
- Technology specific cells in the VaultIP shell module.
  - Memories
    - Mailbox RAM (2 instantiations, one input mailbox and one output mailbox)
    - Program ROM
    - OTP wrapper
    - Data RAM
  - Clock gates
  - FRO cells and related components / standard cells
- Firmware FW1.1.4

  - VaultIP Firmware. Located inside the Program ROM.
    This includes token management from and to the mailboxes (external interface), higher-level crypto operations, key generation, DRBG engine (using AES and Hash hardware modules), asset management, DMA setup and generic engine control.
      - PKA Firmware
- Interfaces located outside the security boundary, but inside the VaultIP top level.
  - AXI Slave
  - AXI Master
    Interface module converts a DMA request into a data transaction from TCM to external AXI or reverse.

3 Ports and Interfaces

The VaultIP module embeds a single slave and master interface. The slave interface is used to receive commands from one or more host CPUs and send the appropriate response. The master interface is used for autonomous data reads and writes from and to an external memory, flash or interface.

Additionally, VaultIP includes physical ports for showing the crypto module status (e.g. FIPS mode and error bits), establishing the role that is requesting services, and resetting the crypto module.

3.1 Physical ports

The summary of interface pins located on the physical boundary of VaultIP is given in the tables below and shown in Figure 2. For clarity, signals are grouped by function in separate tables. Each port pin provides its name, direction, clock domain and function.

The first set of signals in the table below is hardware related and drives the various clock and reset signals.

Port Name	Direction	Clock Domain	Function
Clocks			
reg_clk	IN	reg_clk	Host interface clock.
ctr_clk	IN	ctr_clk	System counter clock signal. This clock signal may not be gated and must be connected to a fixed frequency, while the other clock speed could vary.
cm_clk	IN	cm_clk	Internal crypto-module clock
Reset			
reg_reset_n	IN	reg_clk	Host interface reset
ctr_reset_n	IN	ctr_clk	Counter reset. This reset signal may only be active (set to ‘0’) when the system is reset and must remain inactive after that, such that the counters remain counting.
cm_reset_n	IN	cm_clk	Module reset
clk_man_reset_n	IN	n/a	This signal provides a means to reset all flip-flops inside the clock gate modules, e.g. for DFT and for the simulation purposes to start in a known state. The clock gates are typically not connected to the global reset_n signal because it may be required to have the clocks running during a system reset.
External clock signals for dynamic clock control			
reg_clk_busy	OUT	cm_clk	Indicates that the host interface is busy with host bus transfers. When 1b indicates active transfer on host bus
ctr_clk_busy	OUT	ctr_clk	When 1b, indicates that the counter clock domain is active. This signal is always asserted (set to ‘1’), except when the counter module is in reset (ctr_reset_n set to ‘0’).
cm_clk_busy	OUT	cm_clk	When 1b, indicates that the module is active and busy with processing data and tokens.

Table 2 - Clock and Reset ports

The second group is related to software reset and is designed only for testing purposes: this functionality is tied-off in the production cryptographic module and they are only provided for reference[1].

Port Name	Direction	Clock Domain	Function
soft_reset [For FIPS: tied-off to zero: 1'b0]	IN	reg_clk	Soft reset input. When this signal is made high, internal (state) registers is cleared and crypto engines are reset.
abort_req [For FIPS: tied-off to zero: 1'b0]	IN	reg_clk	Abort request signal. A high level of this signal indicates a request for a soft reset of the *VaultIP* module.
abort_ack [For FIPS: unconnected]	OUT	reg_clk	Abort acknowledge signal. A high level of this signal indicates that the *VaultIP* module is ready to receive a soft reset.

Table 3 - Soft reset ports

The third group provides signals to indicate the status of VaultIP:

Port Name	Direction	Clock Domain	Function
FIPS status signals			
fips_mode	OUT	cm_clk	When active (set to '1'), VaultIP is in FIPS mode. The value of this bit equals the value of the corresponding bit in the MODULE_STATUS register (bit 0).
nonfips_mode	OUT	cm_clk	When active (set to '1'), VaultIP is in non-FIPS mode. The value of this bit equals the value of the corresponding bit in the MODULE_STATUS register (bit 1).
fatal_error	OUT	cm_clk	If asserted (set to '1'), VaultIP detected a fatal error and stops operation. Fatal errors can happen when the CRC on the Firmware ROM fails or when a self test fails. The value of this bit equals the value of the corresponding bit in the MODULE_STATUS register (bit 31).
Secure Debug Control Outputs			
debug_ctrl_out[7:0]	OUT	cm_clk	Secure debug control output bus

Table 4 - Status Signal ports

The following table provides the signals of the target Host interface. Besides global configuration, this physical interface provides the token interface to the mailboxes. Writing to the mailbox triggers processing. After processing, the results can be read from the output mailbox using this same physical interface.

[1] VaultIP can be completely reset by a hard reset. The firmware can be reset using a reset token, provided via the Control Input Interface.

Port Name	Direction	Clock Domain	Function
cpu_id_int[2:0]	IN	reg_clk	Processor *Host* ID bits. These bits must be valid together with **dr_ehi_cs**.
dr_ehi_cs	IN	reg_clk	Select. Selects the slave interface for a bus transfer.
dr_ehi_nrw	IN	reg_clk	Write. Indicates the direction of a bus transfer.
dr_ehi_wbl[3:0]	IN	reg_clk	Write Enable. When a bit is HIGH, and asserted with **dr_ehi_nrw**, it means the corresponding data byte must be written. *VaultIP* only support 32-bit word transfers, therefore this signal bus must be forced to all ones when **dr_ehi_nrw** is asserted high.
prot_acc_n	IN	reg_clk	This bit indicates an access from the Crypto Officer role (0b) or the User role (1b). In VaultIP, this signal is connected to bit [1] of the 3-bit PROT signal of the AXI bus interface.
dr_ehi_addr[13:0]	IN	reg_clk	Address bus. Indicates the target address for the bus transfer (byte addressable).
dr_ehi_wd[31:0]	IN	reg_clk	Write Data bus. Transfers data from Master to Slave.
dr_ehi_rd[31:0]	OUT	reg_clk	Read Data bus. Transfers data from Slave to Master.
slave_ifc_busy	OUT	reg_clk	Ready Output. Indicates extension of the bus transfer, when asserted (‘1’).
reg_reset_n_ifc	OUT	reg_clk	Slave interface reset, aligned with the VaultIP internal register interface reset.

Table 5 - Slave interface ports (Host processor bus)

The next table provides the signals of the master data interface. This physical interface provides DMA signals and a target TCM interface. A request for data is initiated via the DMA interface. It is a requirement from the external system to provide the requested data on the TCM interface. Depending on the indicated direction of the DMA, input data is requested, or result data is available to be read.

Port Name	Direction	Clock Domain	Function
DMA interface			
ext_dma_error	IN	cm_clk	Indicates an external DMA bus error occurred; the transfer must be considered as failed and the operation must be aborted with an error.
ext_dma_control[7:0]	OUT	cm_clk	The DMA control output defines the properties of the DMA. The values of these bits reflect the statically programmed DMA options. Bits 7..4 Reserved, always zero. Bit 3 MST1_SWAP Bits 2:0 MST1_BURST_SIZE
ext_dma_req	OUT	cm_clk	Request. If asserted high, the *VaultIP* module requests a DMA operation. This signal is de-asserted to zero (‘0’) when ext_dma_grant is asserted (‘1’). Only when a DMA operation is completed, a new request is done.

Port Name	Direction	Clock Domain	Function
ext_dma_grant	IN	cm_clk	Grant. This signal grants the DMA request. When both ext_dma_grant and ext_dma_req are active at the same time, the DMA is handshaked.
ext_dma_direction	OUT	cm_clk	Direction, indicates the direction of the DMA operation. A one indicates a read from external data source, writing to the *VaultIP* TCM port. A zero indicates a read from the *VaultIP* TCM port, writing to an external data source. This signal is valid when ext_dma_req is asserted (set to '1').
ext_dma_source_addr [47:0]	OUT	cm_clk	This bus provides the DMA source address. If the ext_dma_direction is zero, this bus provides the *VaultIP* TCM address, if ext_dma_direction is one, this bus provides the external memory address. The value on the bus is valid when ext_dma_req is asserted (set to '1').
ext_dma_dest_addr[47:0]	OUT	cm_clk	This bus provides the DMA destination address. If the ext_dma_direction is one, this bus provides the *VaultIP* TCM address, if ext_dma_direction is zero, this bus provides the external memory address. The value on the bus is valid when ext_dma_req is asserted (set to '1').
ext_dma_prot	OUT	cm_clk	Reflects the protection bit of the DMA operation. This is a copy of Bit 6 of the internal DMA configuration register (AXI_MST1_PROT[1]), programmed by the firmware for each individual operation. This signal is valid when ext_dma_req is asserted (set to '1').
ext_dma_length[20:0]	OUT	cm_clk	Length. Provides the DMA length in bytes. The value on the bus is valid when ext_dma_req is asserted (set to '1').
ext_dma_xfer_active	IN	cm_clk	Indicates that the master bus interface is busy executing the DMA request. It must be asserted shortly after accepting the DMA request signal. This signal must be deasserted when the master bus interface completed the DMA transfer.
DMA TCM Port			
dmac_tcm_cs	IN	cm_clk	Select. Selects the slave interface for a bus transfer.
dmac_tcm_write	IN	cm_clk	Write. Indicates the direction of a bus transfer.
dmac_tcm_wbl[3:0]	IN	cm_clk	Write Enable. When a bit is HIGH, and asserted with dmac_tcm_write, it means the corresponding data byte must be written. If dmac_tcm_wbl[0] is asserted, dmac_tcm_wdata[7:0] is valid and must be written. If dmac_tcm_wbl[1] is asserted, dmac_tcm_wdata[15:8] is valid and must be written. If dmac_tcm_wbl[2] is asserted, dmac_tcm_wdata[23:16] is valid and must be written. If dmac_tcm_wbl[3] is asserted, dmac_tcm_wdata[31:24] is valid and must be written.
dmac_tcm_addr[16:0]	IN	cm_clk	Address bus. Indicates the target address for the bus transfer (byte addressable).

Port Name	Direction	Clock Domain	Function
dmac_tcm_wdata[31:0]	IN	cm_clk	Write Data bus. Transfers data from Master to Slave.
dmac_tcm_rdata[31:0]	OUT	cm_clk	Read Data bus. Transfers data from Slave to Master.
dmac_tcm_wait	OUT	cm_clk	Ready Output. Indicates extension of the bus transfer, when asserted ('1').
cm_reset_n_ifc	OUT	cm_clk	Master interface reset, aligned with the VaultIP internal module reset.

Table 6 - Master DMA / TCM interface ports

The next table provides an overview of the interrupt outputs. By default, the enabled number of interrupts equals the number of mailboxes. If the slave interface supports secure accesses, host 0 has a dedicated IRQ for that. When accessing the *VaultIP* module securely (prot_acc_n = 0b), it can also configure the mailboxes and interrupts. During integration, other optional interrupts can be enabled in the module, dependent on the system host CPUs and access requirements to *VaultIP*.

Port Name	Direction	Clock Domain	Function
host_0_sec_irq	OUT	reg_clk	Combined interrupt output (active HIGH) for one or more Hosts and Domain. Represents Host0, secure interrupt. The interrupt controller is only accessed when Host ID equals 0 and PROT is zero (secure). This interrupt is only available when the slave interface has a prot signal.
host_0_irq	OUT	reg_clk	Combined interrupt output (active HIGH) for one or more Hosts and Domain. Represents Host0, non-secure interrupt. The interrupt controller is only accessed when Host ID equals 0 and PROT is one (non-secure).
host_1_irq	OUT	reg_clk	Combined interrupt output (active HIGH) for one or more Hosts and Domain. Represents Host1, non-secure interrupt. The interrupt controller is only accessed when Host ID equals 1 and PROT is one (non-secure).
host_2_irq	OUT	reg_clk	Combined interrupt output (active HIGH) for one or more Hosts and Domain. Represents Host2, non-secure interrupt. The interrupt controller is only accessed when Host ID equals 2 and PROT is one (non-secure).

Table 7 - Interrupt signal ports

The final set of signals available on the FIPS boundary is the SCAN and FRO characterization signals. Only the *scan_mode_en* and *tst_fro_iddq* signals are connected for device production test purposes. The direct FRO input and output characterization signals are tied-off (inputs) or left unconnected (output)[2].

The FRO test signals that are tied-off in the FIPS-FPGA design, are available on the module boundary for a customer that wants to characterize the FROs. Typically this is done in a test device only. Production devices typically tie these signals off, or disable them after characterization before the module is actually used.

Port Name	Direction	Clock Domain	Function
Characterization / FRO Characterization			
scan_mode_en	IN	None	Active HIGH enable signal for scan_test_mode. This signal typically comes from a testmode controller and is used to break unwanted combinatorial loops during scan test.
tst_fro_iddq	IN	None	Active HIGH enable signal for IDDq testing – this forces all fro_enable outputs LOW to shut down all FROs, overruling all other control signals. This is a combinatorial function, TRNG module clocks don't need to run for this to work.
tst_fro_ctrl_en <= 1'b0	IN	None	Active HIGH enable signal for FRO characterization (enables the tst_fro_select, tst_fro_enable and tst_fro_delay inputs). This is a combinatorial function, TRNG module clocks don't need to run for this to work.
tst_fro_select[4:0] <= 5'b00000	IN	None	FRO selection input (valid values 0-7). A selected FRO will have its fro_testin input forced LOW.
tst_fro_enable <= 1'b0	IN	None	Active HIGH enable signal for FRO selected by tst_fro_select.
tst_fro_delay <= 1'b0	IN	None	Delay chain length selection for FRO selected by tst_fro_select. This input should only be changed while tst_fro_enable is LOW.
tst_fro_clk_out (unconnected)	OUT	None	Output clock signal on the FRO shell module from the FRO selected by tst_fro_select, forced 'low' when tst_fro_ctrl_en is 'low'. This output can also be activated for register-controlled characterization.

Table 8 - TRNG control ports

3.2 Logical Interfaces

The Slave Interface ports communicate with the host through the AXI slave interface, providing the token interface to the mailboxes. Writing to the mailbox triggers processing. Once the input token is processed completely, the results can be read from the output mailbox using this same interface. The Firmware running on the embedded sequencer reads the input tokens, starts processing and writes the output token triggering the corresponding interrupt. Based on the interrupt, an external host can read the result output token. Output is not available in the output mailbox until the input token is fully processed.

[2] *VaultIP* provides a token that can be used to return sampled TRNG outputs. The requested number of bits is returned over the *VaultIP* data output interface using DMA.

Input tokens sent through the Slave Interface constitutes data input and control input interfaces; output tokens constitutes data output and status output interfaces.

The Master DMA/TCM interface communicates with the host through the AXI master interface, providing DMA signals and a target TCM interface. A request for data is initiated via the DMA interface. It is a requirement from the external system to provide the requested data on the TCM interface. Depending on the direction of the DMA, input data is requested, or result data is available to be read.

The Master DMA/TCM interface provides support for data input and data output interfaces.

VaultIP also includes additional ports for control input and status output:

- The Clock and Reset ports provide Control Input and Status Output interfaces.
- The Soft Reset ports (testing only) provide Control Input and Status Output interfaces.
- The Status Signal ports provide the Status Output interface.
- Interrupt signal ports provide the Status Output interface.
- The TRNG control ports (testing only) provide Control Input and Status Output.

Finally, the power port of the FPGA used for the validation provides power input to VaultIP.

The following table shows the mapping between the ports available in VaultIP and the logical interfaces:

Port Groups	Data Input	Data Output	Control Input	Status Output	Power Input
Clock and Reset ports			✓	✓	
Soft Reset ports			✓	✓	
Status Signal ports				✓	
Slave interface ports (AXI Slave Interface)	✓	✓	✓	✓	
Master DMA / TCM interface ports (AXI Master Interface)	✓	✓			
Interrupt signal ports				✓	
TRNG control ports			✓	✓	
FPGA power port					✓

Table 9 - Logical Interfaces

4 Roles, Services and Authentication

4.1 Roles

The VaultIP module is usually installed as part of a System on Chip (SoC), where applications running on the system can use the VaultIP cryptographic services. Applications must identify and authenticate to VaultIP through one of the following roles:

- **User Role**: This role performs general services, cryptographic operations, and asset management functions.
- **Crypto Officer Role**: This role can perform the same functionality as the user role, but it also performs initialization services in the cryptographic module (e.g. configuration of the TRNG engine, write operations in OTP and registers).

The VaultIP module implements a role-based authentication method. See section 4.3 for a description of the Identification and Authentication mechanism implemented in VaultIP.

Internal mechanisms of the VaultIP ensure that User and Crypto Officer service requests and assets (key material and other CSPs) are properly separated and protected.

Next section shows the services provided by VaultIP and the roles that can request them. All services require an authorized role.

4.2 Services

The following table presents the services provided by VaultIP, including:

- The input token through which the service is requested.
- The authorized roles: a checkmark in the role column indicates that the user authenticated with that role can perform the service.
- The cryptographic algorithms involved in the service. The algorithm can be split in several roles for different modes or key lengths as they may not be available in FIPS mode of operation.
- Whether the service is available in FIPS mode (actually, all services are available in FIPS mode of operation, however, some services including cryptographic algorithms or key lengths that are not approved are specifically discriminated in more than one row).
- The Keys and Critical Security Parameters (CSPs) involved in the service.
- How the service accesses the Keys and CSPs: (C)reate (create and fill the CSP), (R)ead (read an existing CSP), (U)pdate (write an existing CSP), (D)elete (delete and zeroize memory where the CSP was stored). An asterisk (*) indicates that the CSP is transported via the token or the Host memory DMA (assets cannot be used).

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
Data copy using DMA	NOP	✓	✓				

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
Encryption and Decryption	Encryption	✓	✓	AES (ECB, CBC, CTR)	✓	AES key IV Counter	R RU RU
		✓	✓	AES (CCM)	✓	AES key Nonce Tag (encrypt) Tag (decrypt)	R R* C* R*
		✓	✓	AES (XTS)	✓	AES key	R
		✓	✓	AES (f8)		AES key IV f8 SaltKey f8 IV	R R*C* R* R*
		✓	✓	Triple-DES (ECB, CBC)	✓	Triple-DES keys IV	R RU
		✓	✓	DES (ECB, CBC)		DES key	R*
Message Digest	Hash	✓	✓	SHA-1	✓	TempDigest[3]	RU
		✓	✓	SHA-224, SHA256	✓	TempDigest[3]	RU
MAC Generation	MAC	✓	✓	HMAC-SHA-1		TempMAC[4] HMAC key	RU R
		✓	✓	HMAC-SHA-224, HMAC-SHA-256	✓	TempMAC[4] HMAC key	RU R
		✓	✓	AES-CMAC	✓	TempMAC[4] AES key	RU R
		✓	✓	AES-CBC-MAC		TempMAC[4] AES key	RU R
MAC Verification	MAC	✓	✓	HMAC-SHA-1	✓	TempMAC[4] HMAC key	RU R
		✓	✓	HMAC-SHA-224, HMAC-SHA-256	✓	TempMAC[4] HMAC key	RU R

[3] When input data to be hashed exceeds 2^{21} - 64 bytes, an intermediate digest (TempDigest) is stored internally.
[4] When input data to be hashed exceeds 2^{21} - 64 bytes, an intermediate MAC (TempMAC) is stored internally.

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
		✓	✓	AES-CMAC	✓	TempMAC[4] AES key	RU R
		✓	✓	AES-CBC-MAC		TempMAC[4] AES key	RU R
ECDH/ECDSA key check	Public Key	✓	✓	ECDH	✓	EC parameters ECDH private key *(opt)* ECDH public key *(opt)*	R R R
		✓	✓	ECDSA	✓	EC parameters ECDSA private key *(opt)* ECDSA public key *(opt)*	R R R
ECDSA Sign	Public Key	✓	✓	ECDSA (P-192), SHA-1, SHA-224, SHA-256		EC parameters ECDSA private key TempDigest[3]	R R R
		✓	✓	ECDSA (P-224 to P-521), SHA-224, SHA-256	✓	EC parameters ECDSA private key TempDigest[3]	R R R
ECDSA Verify	Public Key	✓	✓	ECDSA (P-192 to P-521), SHA-1, SHA-224, SHA-256	✓	EC parameters ECDSA public key TempDigest[3]	R R R
RSA Sign	Public Key	✓	✓	RSA-PSS (n=1024, 1536), SHA-1, SHA-224, SHA-256		RSA-PSS private key TempDigest[3]	R R
		✓	✓	RSA-PSS (n=2048, 3072), SHA-224, SHA-256	✓	RSA-PSS private key TempDigest[3]	R R
RSA Verify	Public Key	✓	✓	RSA-PSS (n=1024 to 3072), SHA-1, SHA-224, SHA-256	✓	RSA-PSS private key TempDigest[3]	R R
RSA Sign	Public Key	✓	✓	RSA-PKCS#1-v1.5 (n=1024, 1536), SHA-1, SHA-224, SHA-256		RSA-v1.5 private key TempDigest[3]	R R
		✓	✓	RSA-PKCS#1-v1.5 (n=2048, 3072), SHA-224, SHA-256	✓	RSA-v1.5 private key TempDigest[3]	R R
RSA Verify	Public Key	✓	✓	RSA-PKCS#1-v1.5 (n=1024 to 3072), SHA-1, SHA-224, SHA-256	✓	RSA-v1.5 private key TempDigest[3]	R R

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
ECDH/ECDSA generate public key	Public Key	✓	✓	ECC multiply (P-192)		ECDH/ECDSA private key EC parameters ECDH/ECDSA public key	R R U
ECDH/ECDSA generate public key	Public Key	✓	✓	ECC multiply (P-224 to P-521)	✓	ECDH/ECDSA private key EC parameters ECDH/ECDSA public key	R R U
ECDH/ECDSA generate private and public key	Public Key	✓	✓	ECC multiply (P-192), CTR_DRBG		ECDH/ECDSA private key EC parameters ECDH/ECDSA public key Internal DRBG state Internal Counter DRGB output value	U R U
		✓	✓	ECC multiply (P-224 to P-521), CTR_DRBG	✓	ECDH/ECDSA private key EC parameters ECDH/ECDSA public key Internal DRBG state Internal Counter DRGB output value	U R U
ECDH generate shared secrets (single key-pair)	Public Key	✓	✓	ECDSA (P-192), SHA-256, CTR_DRBG		ECDH private key EC parameters ECDH public key ECDH shared secrets Internal DRBG state Internal Counter DRGB output value	R R R U
		✓	✓	ECDSA (P-224 to P-256), SHA-256, CTR_DRBG	✓	ECDH private key EC parameters ECDH public key ECDH shared secret Internal DRBG state Internal Counter DRGB output value	R R R U
ECDH generate shared secrets (dual key-pair)	Public Key	✓	✓	ECDH (P-192), SHA-256, CTR_DRBG		ECDH private keys (2x) EC parameters ECDH public keys (2x) ECDH shared secret Internal DRBG state Internal Counter DRGB output value	R R R U

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
		✓	✓	ECDH (P-224 to P-256), SHA-256, CTR_DRBG	✓	ECDH private keys (2x) EC parameters ECDH public keys (2x) ECDH shared secret Internal DRBG state Internal Counter DRGB output value	R R R U
AES Key wrapping	AES (Un)Wrap	✓	✓	AES Key Wrap with or without padding as specified in [SP800-38F] AES-CMAC AES-CTR	✓	AES key	R/R*
AES Key unwrapping using	AES (Un)Wrap	✓	✓	AES Key Unwrap with or without padding as specified in [SP800-38F] AES-CMAC AES-CTR	✓	AES key	R*
TRNG Configuration	TRNG Configuration		✓		✓		U U
Get TRNG Random Number	TRNG Get Random Number	✓	✓	TRNG	✓	Raw random data to seed DRBG	C*
Get DRBG Random Number	TRNG Get Random Number	✓	✓	SHA-256, CTR_DRBG	✓	Seed Internal DRBG state Internal Counter DRGB output value	RU RU C*
TRNG Post-processing Verification		✓	✓	SHA-256, CTR_DRBG	✓	(CSPs preserved)	
TRNG Hardware Self-test Verification		✓	✓		✓	(CSPs preserved)	
Asset Create	Asset Create	✓	✓		✓	Asset	C
Static Asset Search	Static Asset Search	✓	✓		✓		

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
Asset Load (derive)	Asset Load	✓	✓	HMAC-SHA-256, AES-CMAC	✓	Asset Key Derivation Key (KDK)	U R
Asset Load (import)	Asset Load	✓	✓	AES-CMAC, AES-CTR	✓	Asset Key Encryption Key (KEK)	U R
Asset Load (AES Key (Un)Wrap)	Asset Load	✓	✓	AES Key (Un)Wrap	✓	Asset AES Trusted-Wrap Key	U R
Asset Load (random)	Asset Load	✓	✓	CTR_DRBG, AES-CMAC, AES-CTR	✓	Internal DRBG state Internal Counter Asset Key Encryption Key (KEK)	U U U R
Asset Load (plaintext)	Asset Load	✓	✓	AES-CMAC, AES-CTR	✓	Asset Key Encryption Key (KEK)	U R
Asset Delete	Asset Delete	✓	✓		✓	Asset	D
Public Data Read		✓	✓		✓	Asset	R*
Monotonic Counter Read	Monotonic Counter Read	✓	✓		✓	Asset	R*
Monotonic Counter Increment	Monotonic Counter Increment	✓	✓		✓	Asset	RU
OTP Data Write	OTP Write		✓	AES-CMAC, AES-CTR	✓	Asset AES Provisioning Key	CU R
Timer Start, Stop, Read	Secure Timer	✓	✓		✓	Asset	CUD
System Information	System Information	✓	✓		✓		
On Demand Self-Tests	Self-Test	✓	✓	All relevant FIPS algorithms	✓	Asset Store	D
Module Reset	Reset	✓	✓	All relevant FIPS algorithms	✓	Asset Store	D
Authenticated Unlock Start	Authenticated Unlock Start	✓	✓	CTR_DRBG, SHA-256	✓	Authentication Key Authentication State	R U
Authenticated Unlock Verify	Authenticated Unlock Verify	✓	✓	RSA-PKCS#1-v1.5 SHA-256 (n=2048, 3072)	✓	Authentication Key Authentication State	R RU

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
Activate / Deactivate Debug port signals	Set Secure Debug	✓	✓		✓	Authentication Key Authentication State	R R
Register Read	Register Read	✓	✓		✓		
Register Write	Register Write		✓		✓		
Clock Switch	Clock Switch		✓				
Zeroize Output Mailbox	Zeroize Output Mailbox	✓	✓		✓	CSPs in output mailbox	D
Create / Update User Identity	Define Users		✓		✓	User Identities	
Select One-Time-Programmable Zeroize	Select One-Time-Programmable Zeroize		✓		✓	Enable forced zeroize access to the CSPs in OTP for the 'Zeroize One-Time-Programmable' service	
Zeroize One-Time-Programmable	Zeroize One-Time-Programmable		✓		✓	CSPs in the OTP and Asset Store	D

Table 10 - VaultIP Services

The following table shows the FIPS-Approved and FIPS-Allowed algorithms, which can be used in services in FIPS mode of operation:

Algorithm	Usage	Key lengths	Modes	CAVS Certificate Number	Standards
AES	Encryption Decryption	128, 192, 256 bits	ECB, CBC, CTR	#2847	[FIPS197] [SP800-38A]
	Encryption Decryption		CCM	#2847	[SP800-38C]
	Encryption Decryption		XTS	#2847	[SP800-38E]
	MAC		CMAC	#2847	[SP800-38B]
Triple-DES	Encryption Decryption	168 bits	ECB, CBC	#1702	[SP800-67] [SP800-38A]
SHA-1	Message Digest			#2389	[FIPS180-4]

SHA-224, SHA-256	Message Digest			#2389	[FIPS180-4]
HMAC-SHA-1	MAC Verification	112-512 bits		#1787	[FIPS198-1]
HMAC-SHA-224, HMAC-SHA-256	MAC Generation MAC Verification	112-512 bits 128-512 bits		#1787	[FIPS198-1]
RSA	Signature Verification	n=(1024 to 3072)	RSA-PSS (no CRT)	#1488	[PKCS#1]
	Signature Generation	n=(2048 to 3072)	RSA-PSS (no CRT)	#1488	[PKCS#1]
	Signature Verification	n=(1024 to 3072)	RSA-PKCS#1v1.5 (no CRT)	#1488	[PKCS#1]
	Signature Generation	n=(2048 to 3072)	RSA-PKCS#1v1.5 (no CRT)	#1488	[PKCS#1]
AES Key Wrapping	Key Wrapping	128, 192, 256 bits	ECB, with and without padding.	as described in SP 800-38F	[SP800-38F] [RFC3394] [RFC5649]
AES Key Wrapping with AES-CMAC, AES-CTR	Key Wrapping	512 bits (2x256-bit AES keys)		#2847	[SP800-38B] [SP800-38A]
Key-based Key Derivation Function (KBKDF)	Key Derivation	128-512 bits	HMAC-SHA-256,	#25	[SP800-108]
ECDSA	Signature Verification	P-192 to P-521 bits		#502	[FIPS186-4]
	Signature Generation Key Generation	P-224 to P-521 bits		#502	[FIPS186-4]
ECDH	Key Agreement	P-224 to P-256 bits	SHA-256	#46 (KAS) #269 (ECC CDH Primitive)	[SP800-56A]
CTR_DRBG	Random Number Generation		AES-256 in CTR mode, no derivation function, prediction resistant disabled	#500	[SP800-90A] [SP800-38A]

Table 11 - FIPS approved and allowed cryptographic algorithms

The following table shows the cryptographic algorithms and their key lengths that are not FIPS-Approved:

Algorithm	Usage	Key lengths	Modes	Standards
AES	Encryption	128, 192, 256 bits	f8	

Algorithm	Usage	Key lengths	Modes	Standards
	Decryption			
DES	Encryption Decryption	56 bits	ECB, CBC	[FIPS46-3]
HMAC-SHA-1	MAC Verification	< 112 bits		[FIPS198-1]
AES-CBC-MAC	MAC	128, 192, 256 bits	CBC	
RSA	Signature Generation	1024, 1536 bits	RSA-PSS (no CRT) RSA-PKCS#1-v1.5 (no CRT)	[PKCS#1]
ECDSA	Signature Generation, Key Generation	P-192 bits		[FIPS186-4]
ECDH	Key Agreement	P-192 bits		[SP800-56A]
TRNG	Random Number Generation			

Table 12 - Non-FIPS approved cryptographic algorithms

These algorithms can only be used in non-FIPS mode of operation, with the exception of the TRNG, which is used to seed and reseed the FIPS-Approved DRBG.

VaultIP also supports the following algorithms for firmware and data integrity:

Algorithm	Usage
CRC-24	Firmware Integrity
CRC-32	Asset Integrity

Table 13 - Other algorithms

When a CSP is loaded or updated in the asset store, VaultIP generates a CRC-32 checksum. Whenever the CSP is used by a service (referenced through its asset ID), VaultIP verifies the integrity of the CSP comparing the existing and re-calculated checksums.

4.3 Identification and Authentication

Role identification is indicated through the use of a single bit hardware signal (*prot_acc_n*) which is provided as side band information for all services. When the input token is written (the token data is available on the data input bus) the sideband signal must be valid.

Role authentication is performed through the use of a 32-bit identity value provided in the input token for all services.

- VaultIP requires the identification and authentication of the user role for each service request; role identification and authentication status is not internally maintained in the cryptographic module. The Crypto Officer role is explicitly selected using the role selection bit (*prot_acc_n=0b*). For each individual token received with the Crypto Officer role selected, VaultIP compares the *32-bit ID field* provided in the input token with a predefined 32-bit value stored inside the OTP of VaultIP (this identity value is pre-defined by vendor and loaded to OTP during the module initialization). If the comparison succeeds, then the input token is processed. Otherwise, the service request is rejected indicating the error in the output token.

- Likewise, the User role is explicitly selected using the role selection bit (*prot_acc_n=1b*). For each individual token received with the user role selected, VaultIP compares the *32-bit ID field* provided in the input token with the stored user IDs (up to four) inside VaultIP. If the identity matches one of the stored user IDs, access is granted and the input token is processed. Otherwise, the request is rejected indicating the error in the output token. User role identities are created by the Crypto Officer role and reside in volatile memory.

If the authentication fails, VaultIP waits at least 15ms (minimum value set for the highest chip frequency) before it returns the output token rejecting the service request.

There is only one Crypto Officer role identity; this value is stored in the OTP during device initialization. Once it is stored, the Crypto Officer role identity cannot be changed.

For the User role, the *Define Users* token allows the Crypto Officer to create or modify up to four identities, whose values are stored internally in VaultIP's internal memory. These identities do not survive a power-cycle; the Crypto Officer must define the users again anytime VaultIP is reset or powered-up.

Authentication of the User role is performed only when VaultIP is in FIPS mode. The Crypto Officer role is always authenticated, both in FIPS mode and non-FIPS modes of operation.

4.4 Mechanism and Strength of Authentication

The probability of successfully guessing the Crypto Officer Identity is $1/2^{32}$, whereas the probability of successfully guessing one User Identity is less than or equal to $4/(2^{32})$. In both cases, the FIPS 140-2 requirement of having an acceptance rate of less than 1/1,000,000 is met.

With the 15ms delay for failed authentication, VaultIP can process at most 4000 consecutive failed authentication attempts within one minute (60s / 0.015s). In case of the Crypto Officer identity, the overall success rate is $4000 * 1/2^{32}$ (≤ 1/1,073,741); in the case of the User identity, the overall success rate is less than or equal to $4000 * 4/2^{32}$(≤ 1/268,435). In both scenarios, the FIPS 140-2 requirement of having less than 1/100,000 false acceptance rate within one minute is also met.

4.5 Authentication Data protection

The Crypto Officer Identity is stored in the OTP and cannot be modified. The four identities assigned to users with the User role reside in internal memory and can be only created or updated by the Crypto Officer, who needs to authenticate with the correct Identity.

The user role identities are zeroized when VaultIP is powered-off; the Crypto Officer Identity remains in the OTP but can be zeroized through OTP zeroization.

No authentication data can be output by any of the available services.

5 Physical Security

For the purpose of this validation, VaultIP was synthesized in the XC7Z020-CLG484-1 FPGA (Zynq-7000 all Programmable SoC series). This FPGA is a single chip encased in a standard black, opaque epoxy IC package that prevents any access to the interior of the module and conforms to Level 3 requirements for physical security.

The hardness testing was performed on the module with a high temperature +78 C^o and a low temperature of -10 C^o. The testing had no effect on the physical security or operation of the module.

6 Operational Environment

The module operates in a non-modifiable environment.

7 Cryptographic Key and CSP Management

The Asset Store provides the core mechanism for secure handling of key material and other sensitive data like the IV, digest, MAC and state. The Asset Store is designed to store asset objects and allows them to be used, while never revealing their contents outside of VaultIP. The Asset Store identifies the following three types of asset types:

- *Static Asset*: this type of asset is key material that is available immediately after power-up and is located in the OTP of VaultIP. This asset cannot be modified or output (nevertheless, the whole OTP can be zeroized by the Crypto-Officer, see section 7.5).
- *Dynamic Asset*: this type of asset can be key material or any other sensitive data like the IV, digest, MAC and state. This asset needs to be loaded into the Asset Store before its use because it is located in the internal memory of VaultIP and zeroized before the module is powered-off. This asset cannot be modified, only deleted from the asset store.
- *Public data object*: this type of asset is data that can be retrieved from the Asset Store in plaintext for use outside VaultIP and can reside either in the OTP or the internal memory. This asset cannot be modified, only deleted from the asset store.

Static assets are used in the same way as dynamic assets, the only difference is their lifespan. Both are stored in plaintext within VaultIP and protected from disclosure and modification.

Whenever an asset is loaded or updated, VaultIP generates a CRC-32 checksum for that asset; a read of an asset will verify its integrity re-computing the CRC-32 checksum and comparing it to the stored checksum.

Table 14 summarizes the cryptographic keys used in VaultIP with the key lengths supported, the available methods for key generation, entry and output and the way they are stored. Notice that for Key Entry:

- "Firmware" means that the keys are determined during the delivery process and are unique to a given customer. See section 7.2.2 for more information.
- "KeyBlob" is a block of binary data encrypted through a key wrapping method using the AES-CMAC and AES-CTR algorithms. See section 7.2.1 for more information.

Name	Key Length[5]	Generation			Entry					Storage		Output
		KBKDF	Random	Key Pair	Firmware	OTP	Plaintext	KeyWrap	KeyBlob	Static	Dynamic	
Trusted Root Key	128, 256 bits					✓				✓		
Trusted Key Encryption Key (KEK)	256 bits	✓									✓	
Trusted Key Derivation Key (KDK)	128, 256 bits	✓				✓				✓	✓	
Authentication Key (RSA public key)	1024-3072 bits				✓					✓		
Provisioning Key (AES-CMAC, AES-CTR)	512 bits				✓					✓		
AES key	128, 192, 256 bits	✓	✓				✓	✓	✓		✓	✓[6]
Triple-DES key	192 bits	✓	✓				✓	✓	✓		✓	✓[6]
HMAC key	SHA-1: 80-512 bits SHA-224: 112-512 bits SHA-256: 128-512 bits	✓	✓				✓	✓	✓		✓	✓[6]
RSA key pair	1024-3072 bits						✓	✓	✓		✓	✓[7]
ECDSA key pair	192-521 bits			✓			✓	✓	✓		✓	✓[7]
EC Diffie-Hellman key pair	192-521 bits			✓			✓	✓	✓		✓	✓[7]

Table 14 - Cryptographic Key Life Cycle

All of the CSPs that VaultIP processes are listed in the “Keys and CSPs” column in Table 10. Their creations, updates and usage are tied to the corresponding service(s) as shown in Table 10. They reside in the internal memory of VaultIP and get zeroized when the module is powered off.

The keys in OTP are static and can be zeroized with a dedicated service. The Provisioning Key is a key that is available for OTP initialization only. When the complete OTP is initialized the Provisioning Key cannot be used anymore. The other key that is located in firmware, the Authentication Key, is a public key.

7.1 Key Generation and Key Establishment

VaultIP provides services for generating asymmetric and symmetric keys.

AES, Triple-DES and HMAC symmetric keys can be generated or established through the following methods:

[5] Not all key lengths are allowed in FIPS mode of operation. See section 4.2 for more information.
[6] AES, TDES and HMAC keys (loaded as plaintext or generated randomly) can be output as a Keyblob.
[7] The Private Key of ECDSA and ECDH (loaded as plaintext or generated) can be output as Keyblob whereas the Public Key can be output as plaintext when generated or as a Keyblob when loaded as plaintext. The Private and Public keys of RSA (loaded as plaintext) can be output as Keyblob.

- Key-based Key Derivation Function ([SP800-108])
- Random Number Generation (DRBG [SP800-90A])
- EC Diffie-Hellman ([SP800-56A])

Asymmetric key generation is limited to Elliptic Curve DSA (ECDSA), which also uses the Random Number Generation (DRBG [SP800-90A]). Diffie-Hellman (ECDH) key pairs are generated by ECDSA.

Intermediate key generation values are not output from the cryptographic module during or after processing the service.

7.2 Key Entry and Output

Electronic key entry method is used to import the private/secret keys; there is no manual key import or export method used in VaultIP.

7.2.1 Dynamic assets

Dynamic assets can be input into VaultIP through the following methods:

- Load plaintext from the *Host* (*Asset Load Plaintext* service).
- AES Key Wrapping as specified in NIST SP 800-38F (*Asset Load AES-Wrap* service).
- AES Key Wrapping with AES-CMAC, AES-CTR *(Asset Load Import service).*

Keys can also be initialized in the asset store using the built-in key generation features provided by VaultIP (see section 7.1)

When plaintext or random number loaded assets must survive a power cycle, they must be stored outside the *VaultIP* Module. The Asset Store is able to generate a block of binary data known as a *Key Blob* in which the asset is exported to the *Host*. VaultIP utilizes the AES-CMAC, AES-CTR algorithms to protect the asset inside the *Key Blob* from disclosure and modification, using a Trusted Key Encryption Key (KEK). An important part of the protection provided by the *Asset Store* is to allow a *Key Blob* to be created only once, when the asset is filled using the *Asset Load Plaintext* (key provided in plaintext by the host) or *Asset Load Random* (key generated by VaultIP using the DRBG) services.

7.2.2 Static assets

VaultIP provides functionality to program *static assets* or *public data objects* into the OTP via a special service only available to the Crypto Officer role and using an AES Key Blob using AES-CMAC and AES-CTR that is intended for provisioning. The provisioning Key Blob can be created outside VaultIP using a special shared secret. Ownership is covered through the policy of assets that are intended to be written to OTP.

Static assets stored in OTP cannot be output. The OTP can also contain Public data objects which are accessible through its identifier.

Additionally, VaultIP includes two types of keys that are stored in the Firmware and can be input during the integration of the cryptographic module:

- Three Authentication Keys (RSA public key) of 1024, 2048 and 3072 bits.
- One 512-bit AES Provisioning Key (AES-CMAC, AES-CTR)

Note: The program ROM area for these keys in the Firmware is not considered for the integrity verification of the Firmware component of VaultIP.

7.3 Key access control and usage

VaultIP manages the concept of asset ownership and usage policy based on the following information provided during asset creation:

- *Host ID*: host that owns the asset
- *Protection bit*: indicates whether the user is a Crypto Officer or User role
- *Identity*: user ID that owns the asset
- *Usage Policy*: defines how the asset may be used (i.e. algorithm, mode and cryptographic operation)

Once is created, the ownership and usage attributes of the key remains until the key is deleted from the asset store or the asset store is zeroized.

7.4 Key Wrapping

As shown in Table 14 and explained in section Dynamic assets, keys can be entered in encrypted form through the following key wrapping methods and key lengths:

- SP 800-38F based AES Key Wrapping with 128, 192 or 256-bit AES keys. Security strength ranges from 128 to 256 bits, according to the AES key length.
- AES Key Wrapping with AES-CMAC and AES-CTR: two 256-bit AES keys (the first 256-bit-key for the AES-CMAC operation and the second 256-bit-key for the AES-CTR operation). It provides 256-bit security strength.

The following table shows the maximum lengths and security strength of the keys that can be imported to and exported from VaultIP using the key wrapping services:

Key	Maximum Length	Security Strength
Trusted Root Key	256 bits	256 bits
Trusted Key Derivation Key (KDK)	256 bits	256 bits
AES key	256 bits	256 bits
Triple-DES key	168 bits	112 bits
HMAC key	256 bits	256 bits
RSA key pair	3072 bits	128 bits
ECDSA key pair	521 bits	256 bits
EC Diffie-Hellman key pair	521 bits	256 bits

Table 15 - Security Strength of Cryptographic Keys

The maximum strength of the key wrapping schemes provided by VaultIP is greater than or equal to the security strength of the keys that need to be wrapped while entering or exiting the module.

Nevertheless, it is the user’s responsibility to use the AES Key Wrapping with an appropriate key size to ensure the FIPS compliance. Using an insufficient AES key size for AES Key Wrapping will reduce the security strength of the wrapped key.

7.5 Key / CSP Zeroization

A dynamic asset (key or CSP) is deleted from the asset store using the *Asset Delete* service and the memory where the asset was stored is zeroized.

Additionally, the whole asset store is zeroized when VaultIP changes the mode of operation from non-FIPS mode to FIPS mode [8], when it transitions to the error state or when the module is powered-off.

Keys and CSPs considered static assets are stored in OTP; VaultIP provides a two-step process to zeroize the OTP memory. First, the service *Select One-Time-Programmable Zeroize* must be called to enable OTP zeroization. After OTP zeroization is enabled, the *Zeroize One-Time-Programmable* service zeroizes the complete OTP (by writing ones to all entries) and the asset store..

VaultIP also provides the *Zeroize Output Mailbox* service can be used to zeroize the output mailbox before the mailbox is unlinked. This operation prevents sensitive material leaking to other Hosts applications.

Zeroization is performed filling the memory area with zeroes. The operation is performed in a time that is not sufficient to compromise CSPs. Additionally:

- VaultIP processes one input message at a time, when a zeroization service (Asset Delete, Zeroize Output Mailbox) is processed no other input message accessing the asset store can be executed.
- When the Self-Test service is executed to switch to FIPS mode of operation, no information is output until VaultIP transitions to the FIPS mode or the error state.
- No information is output when VaultIP transitions to or is in the Error state.

7.6 Random Number Generation

VaultIP includes a Deterministic Random Bit Generator (DRBG) based on the CTR_DRBG algorithm and AES-256 as the underlying cipher according to [SP800-90A]. VaultIP uses this engine to:

- Create symmetric keys in the asset store for the *Asset Load* service.
- Generate asymmetric keys for the *ECDH/ECDSA sign/verify* and *ECDH generate shared secrets* services.
- Randomize projective point with ECC point multiplication.
- Provide random data for the *TRNG Get Random* service.

VaultIP includes a True Random Number Generator (TRNG) engine to provide entropy input to the DRBG. Using the *TRNG Configuration* service, the Crypto Officer can seed or re-seed the DRBG (the service can also start the TRNG engine in the same service).

It is also possible to trigger an automatic re-seed of the DRBG using the same service when requesting sufficient random data; in this case the operation can be performed by both the Crypto Officer and the User roles.

The underlying TRNG provides 384 bits of entropy to seed the DRBG; the DRBG itself provides 256 bits of security strength.

VaultIP performs continuous tests in the DRBG and TRNG engines, verifying that the previous and current generated blocks of random data are not equal.

[8] There is no transition from FIPS mode to non-FIPS mode of operation, VaultIP must be reset to go to the non-FIPS mode of operation.

7.7 True Random Number Generation

The True Random Number Generator (TRNG) engine is a Non-Deterministic Random Number generator (NDRNG) containing a hardware entropy source based on free running ring oscillators. This TRNG utilizes multiple Free Running Oscillators (FROs) to supply the entropy needed to generate true random numbers. The number of FROs in the actual design is 8.

According to [SP 800-90B] the amount of entropy input to a ‘Conditioning Function’ must be twice the amount that is output by that ‘Conditioning Function’ and the 8-bit samples output by the Noise Source provided by the FROs are supposed to have one bit of entropy each. In order to generate 384 bits of entropy to (re-)seed the DRBG, the TRNG requires 6144 sample bits, equivalent to 12 blocks of raw Noise Source input. Note that the granularity of entropy generation is in blocks of 256 bits; three of those blocks are needed for two (re-)seed operations. The 128-bits excess data from the second block is stored in the VaultIP memory to be combined with the third block.

The set of FROs provided in the Verilog RTL is ready for synthesis after instantiating cells from the chosen target technology. However, it is possible to optimize the FROs for the specific target technology such that more entropy is generated. Customers should follow the instructions provided in the VaultIP-130 Integration Manual for this purpose.

Using the *TRNG Configuration* service, the Crypto Officer can configure and start the TRNG engine to initialize the DRBG.

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

According to 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, VaultIP is not subject to EMI/EMC regulations because it is a subassembly that is sold to an equipment manufacturer for further fabrication. That manufacturer is responsible for obtaining the necessary authorization for the equipment with VaultIP embedded prior to further marketing to a vendor or to a user.

9 Self Tests

9.1 Power-Up Tests

After successful initialization of the SoC in which VaultIP is integrated, VaultIP automatically performs power-up tests without user intervention to ensure that the module is not corrupted and that the cryptographic algorithms within the module work as expected.

During the execution of the power-up tests, services are not available and no data output is possible.

If the power-up tests succeed, then VaultIP becomes operational and enters non-FIPS mode with the following status signals:

- nonfips_mode = 1b
- fips_mode = 0b
- fatal_error =0b

If the power-up tests fail, then VaultIP transitions to the Error state and becomes non-operational with the following status signals:

- nonfips_mode = 0b
- fips_mode = 0b
- fatal_error =1b

9.1.1 Integrity tests

VaultIP verifies the integrity of the firmware in ROM using a dedicated 24-bit CRC algorithm. After power-up, VaultIP calculates the CRC of the firmware image and compares it against the last word of the firmware code image. If the CRC verification succeeds, VaultIP continues with the rest of the power-up tests; if the CRC verification fails, the cryptographic module sets the fatal_error signal and transitions into the Error state, becoming non-operational.

9.1.2 Cryptographic algorithm tests

VaultIP performs self-tests on all FIPS-Approved cryptographic algorithms that can be used in FIPS mode of operation. VaultIP uses known answer tests (KAT) for the cryptographic algorithms as shown in the following table:

Cryptographic Algorithm	Test
AES	KAT AES-CBC, 128-bit, encrypt KAT AES-CBC, 128-bit, decrypt KAT AES-CCM, 192-bit, encrypt KAT AES-CCM, 192-bit, decrypt KAT XTS-AES, 256-bit, encrypt KAT XTS-AES, 256-bit, decrypt KAT AES-CMAC, 256-bit
Triple-DES	KAT Triple-DES, 168-bit, encrypt KAT Triple-DES, 168-bit, decrypt
SHA-1, SHA-224	KAT SHA-1 KAT SHA-224

Cryptographic Algorithm	Test
HMAC and SHA-256	KAT HMAC-SHA-256
RSA	KAT RSA 2048-bit (PKCS#1 v1.5), signature generation KAT RSA 2048-bit (PKCS#1 v1.5) signature verification
KAS	KAT for Z computation (NIST P-224) HMAC-SHA-256, AES-CMAC KAT (for Key Derivation Function). AES-CTR-256 DRBG, KAT ECDSA (NIST P-224) signature generation, KAT ECDSA (NIST P-224) signature verification (for Prerequisite Algorithms)
ECDSA	KAT ECDSA (NIST P-224) signature generation KAT ECDSA (NIST P-224) signature verification
DRBG AES-CTR (SP900-80A)	KAT AES-CTR-256 DRBG

Table 16 - Power-Up Tests

If any of the known answer tests fail (the calculated output does not equal the known answer), VaultIP transitions to the Error state and becomes non-operational.

9.2 On-demand self-tests

VaultIP provides the *Self-Test* service to perform self-tests on demand and enter FIPS mode of operation. This service performs the same known answer tests executed during power-up tests.

During the execution of the self-tests, services are not available and no data output is possible.

If the self-tests succeed, the module transitions to FIPS mode of operation with the following status signals:

- nonfips_mode = 0b
- fips_mode = 1b
- fatal_error =0b

If the self-tests fail, the module transitions to the Error state and becomes non-operational with the following status signals:

- nonfips_mode = 0b
- fips_mode = 0b
- fatal_error =1b

The Self-Test service can be performed by both the User role and the Crypto-Officer role.

9.3 Conditional Tests

VaultIP performs conditional tests on the cryptographic algorithms shown in the following table:

Algorithm	Test

Algorithm	Test
ECDSA key generation	Pair-wise consistency test
DRBG AES-CTR (SP800-90A)	Continuous test (previous and current random data are not equal)
TRNG	Continuous test (previous and current random data are not equal)
ECDSA/ECDH	Conditional Tests for Assurances are performed following sections 5.5.2, 5.6.2 and 5.6.3 of [SP800-56A]. ECDSA Pair-wise consistency test, DRBG Continuous test for Conditional Tests on Prerequisite Algorithms.

Table 17 - Conditional Tests

If any of these tests fail, VaultIP transitions to the Error state and becomes non-operational.

9.4 Module status

VaultIP provides different interfaces to show its status:

- The *fips_mode*, *nonfips_mode* and *fatal_error* output signals (equivalent to bits 0, 1 and 31 of the MODULE_STATUS register accessible by hosts) indicate whether the cryptographic module is in FIPS mode of operation, non-FIPS mode of operation or in an Error state, respectively. This information is available to any user.
- The *System Info* service provides the FIPS mode state indication, general hardware and firmware version information and eventual OTP anomalies. This service can be requested by both the User and Crypto-Officer roles.
- The output token returned by VaultIP provides the result of processing the service requested by the User or Crypto-Officer role.

9.5 Error state

When a fatal error occurs, VaultIP transitions to the Error state and becomes non-operational, and the Asset Store is entirely zeroized. The module can transition to this state for the following reasons:

- Failure of the power-up tests (including failure of the integrity test) or on-demand self-tests.
- Failure of the conditional tests.
- Failure of the True Random Number Generator (TRNG).
- DMA errors.

Error indication is given by the *fatal_error* output port (which is equal to the value of bit 31 in the MODULE_STATUS register). The signal is set to 1 when a fatal error occurs.

In the Error state, services are not available and no data output is possible with the exception of the *System Info* and *Reset* services. If the Error state is caused by integrity tests, the *System Info* and *Reset* services cannot be used.

There are two options to clear the Error state and bring VaultIP back to an operational state:

- Reset the module.
- Power-off and power-on the module.

10 Design Assurance

10.1 Configuration Management

Inside Secure uses a configuration management system to store all source code, test tools and verification scripts. For product documentation, a directory structure and labeling convention for filenames and directories are used to maintain documents under configuration management.

10.1.1 Cryptographic Module Identification

VaultIP is uniquely identified by the filenames used to ship the IP core Verilog code and the firmware image. The following convention is used:

VaultIP-<nn>[<o>]_HW<x.y.z> [_(alpha|beta|final)]

VaultIP-<nn>[<o>]_Firmware[_cfg<O>]_FW<a.b.c> [_(alpha|beta|final)]

where:

- <nn> is the "EIP number" in INSIDE Secure's IP catalog for the VaultIP module this number is 130. The VaultIP module RTL name is based on this number and hardware design flow therefore references the module as eip130.
- <o> or <O> is a code for specific configuration options, if any. Please see the Data Sheet or Hardware Reference Manual for a detailed explanation of this code, see the Release Notes to see the configuration details for this delivery.
- <x.y.z> is the hardware version number of the IP in this delivery. The 3rd digit is only used for patches to the original <x.y> version.
- <a.b.c> is the firmware version number included in this delivery. The 3rd digit is only used for patches to the original <a.b> version.
- _(alpha|beta|final) is used to distinguish intermediate drops leading towards the final release. "_final" is omitted if no intermediate drops are done or if it is a generic release.

10.2 Delivery and Operation

VaultIP synthesized in the Xilinx ZynQ XC7Z020 FPGA is a single chip hardware module. The chip is delivered from the vendor via a trusted delivery courier. Upon reception of VaultIP, the customer should verify that the package does not have any irregular tears or openings.

The chip comes preloaded with the following code packages:

- HW: SafeXcel-IP-130s-4-axi32_HW1.1.4_SRC_T10_T12.zip
- FW: SafeXcel-IP-130s-4-axi32_Firmware-cfgA_FW1.1.4_SRC_T10_T12_v2.zip

The Crypto Officer ID, which is embedded in the Firmware, will be provided in the package and the *Define User* service can be used to create the user roles. On power-up, the module automatically performs power-up self-tests; successful completion of the integrity checks within the power-up tests ensures the integrity of VaultIP.

10.3 Guidance

For the purpose of this cryptographic module validation, VaultIP is synthesized in the Xilinx ZynQ XC7Z020 FPGA and validated as a single-chip hardware module. Nevertheless, VaultIP, as a soft IP core in written in RTL code, can also be synthesized in other chips and systems. The resulting single-chip hardware module can take advantage of this validation and be subject to a quick re-validation. The guidance of how much re-validation will be necessary on the resulting hardware module is currently under the development by the CMVP.

INSIDE Secure provides the following documentation for integrators to synthesize VaultIP in their chips:

Document Name	Document Number
VaultIP-130 Crypto Officer and User Manual	007-130110-403
VaultIP 130 Security Module, Integration Manual	007-130110-200
VaultIP-130 Security Module, Hardware Reference Manual	007-130110-201
VaultIP-130 Security Module, Firmware Reference Manual	007-130110-204

11 Mitigation of Other Attacks

The cryptographic module does not implement security mechanisms to mitigate other attacks.

A Appendixes

A.1 Glossary and Abbreviations

AES	Advanced Encryption Specification
ASIC	Application Specific Integrated Circuit
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
CVT	Component Verification Testing
DES	Data Encryption Standard
DMA	Direct Memory Access
DSA	Digital Signature Algorithm
FIPS	Federal Information Processing Standards Publication
FPGA	Field-Programmable Gate Array
FRO	Free Running Oscillator
FSM	Finite State Model
HMAC	Hash Message Authentication Code
ICM	Integer Counter Mode
IP	(semiconductor) Intellectual Property (core)
IRQ	Interrupt ReQuest
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
MAC	Message Authentication Code
NDF	No Derivation Function
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback
OTP	One-Time-Programmable memory
O/S	Operating System
PD	Prediction Resistance
PP	Protection Profile

PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
RTL	Register Transfer Level
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SLA	Service Level Agreement
SOC	System on a Chip
SSH	Secure Shell
TCM	Tightly Coupled Memory
TDES	Triple DES
TEE	Trusted Execution Environment
UI	User Interface

A.2 References

FIPS46-3	**Data Encryption Standard (DES); specifies the use of Triple DES** October 1999 http://csrc.nist.gov/publications/fips/archive/fips46-3/fips46-3.pdf
FIPS180-4	**Secure Hash Standard (SHS)** March 2012 http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** July 2013 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	**Advanced Encryption Standard** November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed-Hash Message Authentication Code (HMAC)** July 2008 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
PKCS#1	**Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1** February 2003 http://www.ietf.org/rfc/rfc3447.txt
RFC3394	**Advanced Encryption Standard (AES) Key Wrap Algorithm** September 2002 http://www.ietf.org/rfc/rfc3394.txt
RFC5649	**Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm** September 2009 http://www.ietf.org/rfc/rfc5649.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation - Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping**

	December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-56A	**NIST Special Publication 800-56A Revision 2 - Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography** May 2013 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf
SP800-67	**NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** January 2012 http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf
SP800-90A	**NIST Special Publication 800-90A - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** January 2012 http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf
SP800-90B	**NIST Draft Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** August 2012 http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf
SP800-108	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions** October 2009 http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf