VaultIP

FIPS 140-2 Non-Proprietary Security Policy

Document Revision: F

Document Date: June 2022

Prepared by:
atsec information security Corp.
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Rambus Inc.
4453 North First Street, Suite 100
San Jose, CA 95134
Phone: +1-408-462-8000
https://www.rambus.com/

For further information contact: sipsupport@rambus.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 14.

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 Xilinx Zynq XC7Z045 FPGA embedded in a Xilinx ZC706 base board.

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 is synthesized on the Xilinx Zynq XC7Z045 FPGA chip, which belongs to the Zynq-7000 All Programmable SoC series. VaultIP is defined as a sub-chip hardware module validated at security level 2; its embodiment is of the type of single chip.

The following figure shows the Xilinx Zynq XC7Z045 FPGA in which VaultIP has been synthesized and tested.

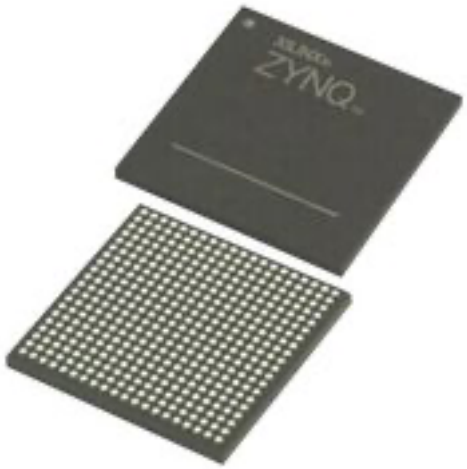

Figure 1 – Xilinx Zynq XC7Z045 FPGA

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

2.4 Approved Modes of Operation

VaultIP has two modes of operation: FIPS mode and non-FIPS mode. The FIPS/non-FIPS modes of operation are implicitly based on the service requested.

- In FIPS mode of operation, only FIPS-Approved or FIPS-Allowed cryptographic algorithms with specific modes and key lengths can be requested. Table 14 shows all algorithms supported by the module in FIPS mode.
- In non-FIPS mode of operation, only the non-approved algorithms can be requested. Table 15 shows the algorithms supported by the module in non-FIPS mode.

VaultIP enforces the separation of keys and CSPs between FIPS mode and non-FIPS mode.

2.5 System Block Diagram

The Figure below 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.

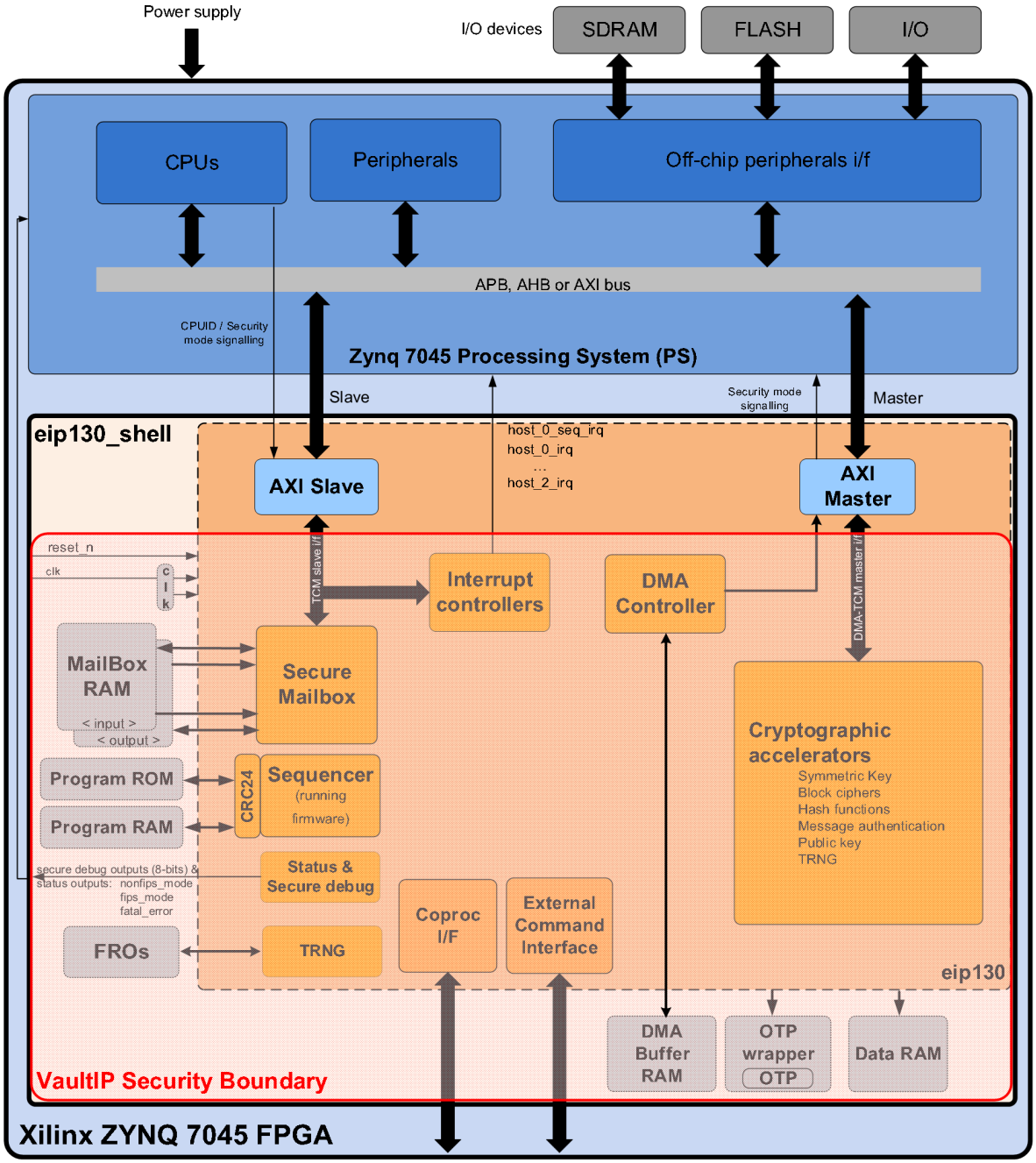

Figure 2 - System Block Diagram

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 and Program RAM). 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.).

For the purpose of this validation, the physical boundary of the module is delimited by the Xilinx Zynq XC7Z045 FPGA (see Figure 1). The ZC706 evaluation board for the XC7Z045 SoC

provides the hardware environment for developing and evaluating the hardware design of VaultIP.

2.6 Hardware Block Diagram

VaultIP consists of two major components, the Verilog RTL and the Firmware running from a ROM and a RAM 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 3; it shows the details of all interfaces that cross the security boundary and the first hierarchy levels of the VaultIP RTL. Please note that for the purpose of this validation, the physical boundary is represented by the FPGA in which the module is synthesized.

Firmware is located in the Program ROM to perform the boot process and in Program RAM once the boot process has completed and VaultIP is in functional mode. 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.

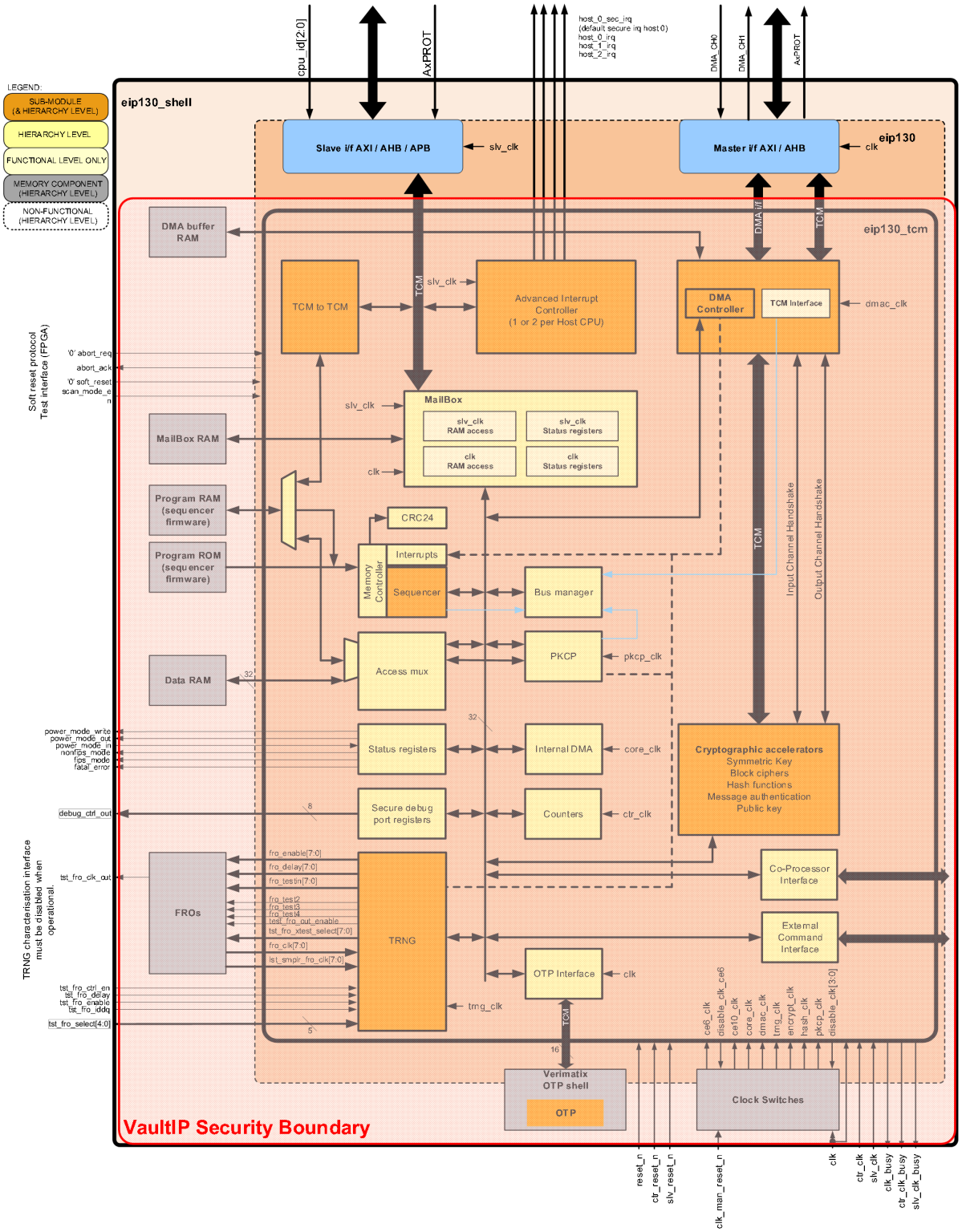

Figure 3 - Hardware Block Diagram

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 HW3.0.3.
  - 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.
    - Triple-DES and DES with ECB and CBC.
    - AES with ECB, CBC, CTR, CMAC, CCM, GCM, 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, SHA-256, SHA-384 and SHA-512
      - 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
    - Program RAM
    - OTP wrapper
    - Data RAM
  - Clock gates
  - FRO cells and related components / standard cells

- Firmware FW3.0.6
  - VaultIP Boot Firmware located in the Program ROM
    This includes code required to securely load & decrypt the main RAM firmware and verify its integrity
  - VaultIP Firmware. Located inside the Program RAM.
    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			
slv_clk	IN	slv_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.
clk	IN	clk	Internal crypto-module clock
Reset			
slv_reset_n	IN	slv_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.
reset_n	IN	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			
slv_clk_busy	OUT	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’).
clk_busy	OUT	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	slv_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	slv_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	slv_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			
fatal_error	OUT	clk	When active (set to '1'), VaultIP detected a fatal error and stops operation. Fatal errors can happen when the CRC on the Firmware Program 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	clk	Secure debug control output bus

Table 4 - Status Signal ports

The fourth group provides signals to indicate the Power mode of VaultIP:

Port Name	Direction	Clock Domain	Function
power_mode_out	OUT	clk	0b: VaultIP-1XX is operational. 1b: VaultIP-1XX is ready to enter 'Sleep Mode'. Only valid in combination with an active power_mode_write signal.
power_mode_write	OUT	clk	When active it indicates the status of power_mode_out is valid.

[1] RoT 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
power_mode_in	IN	clk	The status of this signal is checked by the core during power up after Sleep Mode to see if the state information from Data RAM must be restored. 0b: VaultIP-1XX must not restore the state information from Data RAM after reset. 1b: VaultIP-1XX must restore the state information from Data RAM after reset.

Table 5 - Power Mode Control Signals

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.

Port Name	Direction	Clock Domain	Function
Read Command channel			
cpu_id_rd[2:0]	IN	slv_clk	Read channel processor *Host* ID bit. These bits must be valid together with s_araddr.
s_arprot[2:0]	IN	slv_clk	Read channel protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. The module only uses the Host Secure/Non-Secure bits to accept or ignore accesses to specific asserts and operations.
s_arvalid	IN	slv_clk	Read address valid: When high, this signal indicates that valid read address and control information are available. The address and control information are expected to remain stable until the read address acknowledge signal, s_arready, is asserted.
s_arid[7:0]1	IN	slv_clk	Read address ID: This signal is the identification tag for the read address group of signals.
s_araddr[13:0]2	IN	slv_clk	Read byte address: The read address bus gives the initial byte address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
s_arlen[3:0]	IN	slv_clk	Burst length: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
s_arsize[2:0]	IN	slv_clk	Burst size: This signal indicates the size of each transfer in the burst.
s_arburst[1:0]	IN	slv_clk	Burst type: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
s_arready	OUT	slv_clk	Read address ready: When high, this signal indicates that the AXI slave is ready to accept a read address and associated control signals.

Port Name	Direction	Clock Domain	Function
Write Command channel			
cpu_id_wr[2:0]	IN	slv_clk	Write channel processor Host ID bits. These bits must be valid together with s_awaddr.
s_awprot[2:0]	IN	slv_clk	Write channel protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. The module only uses the Host Secure/Non-Secure bits to accept or ignore accesses to specific asserts and operations.
s_awvalid	IN	slv_clk	Write address valid: When high, this signal indicates that valid write address and control information are available. The address and control information are expected to remain stable until the write address acknowledge signal, s_awready, is asserted.
s_awid[7:0]1	IN	slv_clk	Write address ID: This signal is the identification tag for the write address group of signals.
s_awaddr[13:0]2	IN	slv_clk	Write byte address: The write address bus gives the initial byte address of a write burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
s_awlen[3:0]	IN	slv_clk	Burst length: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
s_awsize[2:0]	IN	slv_clk	Burst size: This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
s_awburst[1:0]	IN	slv_clk	Burst type: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
s_awready	OUT	slv_clk	Write address ready: When high, this signal indicates that the AXI slave is ready to accept a write address and associated control signals.

Table 6 – AXI Slave interface (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
Read Command channel			
dmac_arready	IN	clk / mst_clk	Read address ready. When high, this signal indicates that the AXI slave or interconnect is ready to accept a read address and associated control signals.
dmac_arvalid	OUT	clk /	Read address valid. When high, this signal indicates

Port Name	Direction	Clock Domain	Function
		mst_clk	that valid read address and control information are available from the AXI master. The address and control information must remain stable until the write address acknowledge signal, dmac_arready, is asserted.
dmac_araddr[47:0]1	OUT	clk / mst_clk	Read byte address. The read address bus gives the initial byte address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
dmac_arprot1	OUT	clk / mst_clk	This signal provides the second bit of the protection bus. This signal must drive bit [1] of the arprot bus, representing the Secure/Non-Secure bit. The other two bits may be forced to zeroes.
dmac_arlen[3:0]	OUT	clk / mst_clk	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
dmac_arsize[2:0]	OUT	clk / mst_clk	Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. This will always match the AXI data bus width.
dmac_arburst[1:0]	OUT	clk / mst_clk	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. Either INCR or FIXED, depends on the source address locking DMA parameter for the active DMA Read Channel.
Write Command channel			
dmac_tcm_cs	IN	clk	Select. Selects the slave interface for a bus transfer.
dmac_awready	IN	clk / mst_clk	Write address ready. When high, this signal indicates that the AXI slave or interconnect is ready to accept a write address and associated control signals.
dmac_awvalid	OUT	clk / mst_clk	Write address valid. When high, this signal indicates that valid write address and control information are available from the AXI master. The address and control information will remain stable until the write address acknowledge signal, dmac_awready, is asserted.
dmac_awaddr[47:0]1	OUT	clk / mst_clk	Write byte address. The write address bus gives the initial byte address of a write burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
dmac_awprot1	OUT	clk / mst_clk	This signal provides the second bit of the protection bus. This signal must drive bit [1] of the awprot bus, representing the Secure/Non-Secure bit. The other two bits may be forced to zeroes.
dmac_awlen[3:0]	OUT	clk / mst_clk	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.

Port Name	Direction	Clock Domain	Function
dmac_awsize[2:0]	OUT	clk / mst_clk	Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. This will always match the AXI data bus width.
dmac_awburst[1:0]	OUT	clk / mst_clk	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. Either INCR or FIXED, depends on the destination address locking DMA parameter.
Write Data channel			
dmac_wready	IN	clk / mst_clk	Write ready. When high, this signal indicates that the AXI slave or interconnect can accept the write data.
dmac_wvalid	OUT	clk / mst_clk	Write valid. When high, this signal indicates that valid write data and strobes are available from the AXI master.
dmac_wlast	OUT	clk / mst_clk	Write last. This signal indicates the last transfer in a write burst.
dmac_wstrb[3:0]2	OUT	clk / mst_clk	Write strobes. This signal indicates which byte lanes to update. There is one write strobe for each eight bits of the write data bus. Therefore, dmac_wstrb[n] corresponds to dmac_wdata[(n*8) + 7 : n*8].
dmac_wdata[31:0]2	OUT	clk / mst_clk	Write data. The write data bus is 32 bits wide.
Read Data channel			
dmac_rvalid	IN	clk / mst_clk	Read valid. When high, this signal indicates that the required read data is available from the AXI slave or interconnect and the read transfer can complete.
dmac_rdata[31:0]2	IN	clk / mst_clk	Read data. The read data bus is 32 bits wide.
dmac_rresp[1:0]	IN	clk / mst_clk	Read response. This signal indicates the status of the read transfer.
dmac_rready	OUT	clk / mst_clk	Read ready. When high, this signal indicates that the AXI master can accept the read data and response information.
dmac_bvalid	IN	clk / mst_clk	Write response valid. When high, this signal indicates that a valid write response is available from the AXI slave or interconnect.
Write Response channel			
dmac_bresp[1:0]	IN	clk / mst_clk	Write response. This signal indicates the status of the write transaction.
dmac_bready	OUT	clk / mst_clk	Write response ready. When high, this signal indicates that the AXI master can accept the response information.

Table 7 - 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 coaccess requirements to VaultIP.

Port Name	Direction	Clock Domain	Function
host_0_sec_irq	OUT	slv_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	slv_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	slv_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	slv_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 8 - Interrupt signal ports

The next table provides the signals of the Coprocessor Interface. The Coprocessor Interface allows the VaultIP to be connected to a maximum of 10 external Coprocessors. This interface allows exporting of asset data from the VaultIP to these Coprocessors using the token command. The data will only be exported if the Coprocessor selection bits in the token align with the corresponding bits set in the Asset Policy.

Port Name	Direction	Clock Domain	Function
cp_clk[1]	OUT	**cp_clk**	Coprocessor Interface bus clock. This is the main system clock gated by the cp_clk_dis signal.
cp_clk_dis	OUT	**cp_clk**	Coprocessor Interface bus clock gating control. • When 0'b, enable interface clock. • When 1'b, disable interface clock.
cp_tcm_cs[14:0][2]	OUT	**cp_clk**	These bits select which of the 15 possible external IP's is being accessed. Active HIGH, only one cp_tcm_cs[14:0] bit is set to 1'b at a time.
cp_tcm_wait	IN	**cp_clk**	Active HIGH 'wait' from the external IP's. When set to 1'b by an external IP, the Coprocessor Interface bus output signals will be held stable for a repeated access request in the next clock cycle. External multiplexing will be required if more than one external IP is connected to the bus.
cp_tcm_we[3:0]	OUT	**cp_clk**	Active HIGH byte lane write enables, bit [0] for data LSB bits [7:0], bit [3] for MSB data bits [31:24].
cp_tcm_addr[31:0]	OUT	**cp_clk**	Address of 32-bit word to read/write.

Port Name	Direction	Clock Domain	Function
cp_tcm_wdata[31:0]	OUT	**cp_clk**	Output write data to external IP.
cp_tcm_rdata[31:0]	IN	**cp_clk**	Input data bus from the external IP's. External multiplexing will be required if more than one external IP is connected to the bus.
cp_tcm_err3	IN	**cp_clk**	Active HIGH 'error' from the external IP's. When set to 1'b, signals to EIP-130 that there is an error with the current access. Only sampled if the bus is not being held by cp_tcm_wait. External multiplexing will be required if more than one external IP is connected to the bus.

Table 9 - Coprocessor Interface

Another 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 – these 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.

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

Port Name	Direction	Clock Domain	Function
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 10 - TRNG control ports

The next table provides the signals of the External Command Interface. The ECI allows control of the EIP-130 Secure Debug Control bus, **debug_ctrl_out[7:0]**.

Port Name	Direction	Clock Domain	Function
eci_tcm_cs	IN	cp_clk	External Command Interface select. A high on this pin indicates an access.
eci_tcm_addr[3:0]	IN	cp_clk	External Command Interface 32-bit register address input. Valid when eci_tcm_cs is high.
eci_tcm_we[3:0]	IN	cp_clk	External Command Interface byte write enables. Valid when eci_tcm_cs is high. • eci_tcm_we[0] high for eci_tcm_wd[7:0] write enable. • eci_tcm_we[3] high for eci_tcm_wd[31:24] write enable. A read access is indicated when all write enables are low.
eci_tcm _wd[31:0]	IN	cp_clk	Data to be written to the External Command Interface. Valid when eci_ tcm_cs is high.
eci_tcm_rd[31:0]	OUT	cp_clk	Read data from the External Command Interface.
eci_rd_req	OUT	cp_clk	Read request. When eci_rd_req is high the eci_control register has been updated by the EIP-130. This output is cleared to zero when the eci_control register is read from the ECI port.

Table 11 - ECI Register Interface

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.

Input tokens sent through the Slave Interface constitutes data input and control input interfaces; output tokens constitute 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, data output, control input and status 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.
- The Power Mode Control Signals provide Control Input and Status Output interfaces.
- Interrupt signal ports provide the Status Output interface.
- The TRNG control ports (testing only) provide Control Input and Status Output.
- As an option VaultIP supports a Coprocessor interface. When authorized, this can be used to provide key material to a high-speed external cryptographic engine. The engine is outside the security boundary.
- As an option, VaultIP supports an external control interface or ECI. The ECI may be used to connect a JTAG interface, for example. Use of this port is enabled after a cryptographic challenge / response process.

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				✓	
Power Mode Control signals			✓	✓	
Slave interface ports (AXI Slave Interface)	✓	✓	✓	✓	
Master DMA / TCM interface ports (AXI Master Interface)	✓	✓	✓	✓	
Interrupt signal ports				✓	
Coprocessor Interface	✓	✓			
TRNG control ports			✓	✓	
External Control Interface (ECI)	✓	✓	✓	✓	
FPGA power port					✓

Table 12 - 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.
- The Keys and Critical Security Parameters (CSPs) involved in the service. Note that the general term “Asset” is used to refer to any key or CSP.
- 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	✓	✓			n/a	
Encryption and Decryption	Encryption	✓	✓	AES (ECB, CBC, CTR)	✓	AES key	R
		✓	✓	AES (CCM)	✓	AES key	R
		✓	✓	AES (XTS)	✓	AES key	R

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
		✓	✓	AES (GCM, GMAC)	✓	AES key	R
		✓	✓	AES (f8)		AES key	R
		✓	✓	Triple-DES (ECB, CBC)		Triple-DES keys	R
		✓	✓	DES (ECB, CBC)		DES key	R*
Message Digest	Hash	✓	✓	SHA-1	✓	n/a	
		✓	✓	SHA-224, SHA256	✓	n/a	
		✓	✓	SHA-384, SHA-512	✓	n/a	
MAC Generation	MAC	✓	✓	HMAC-SHA-1		HMAC key	R
		✓	✓	HMAC-SHA-224, HMAC-SHA-256	✓	HMAC key	R
		✓	✓	HMAC-SHA-384, HMAC-SHA-512	✓	HMAC key	R
		✓	✓	AES-CMAC	✓	AES key	R
		✓	✓	AES-CBC-MAC		AES key	R
MAC Verification	MAC	✓	✓	HMAC-SHA-1	✓	HMAC key	R
		✓	✓	HMAC-SHA-224, HMAC-SHA-256	✓	HMAC key	R
		✓	✓	HMAC-SHA-384, HMAC-SHA-512	✓	HMAC key	R
		✓	✓	AES-CMAC	✓	AES key	R
		✓	✓	AES-CBC-MAC		AES key	R
ECDH key check	Public Key	✓	✓	ECDH		EC parameters ECDH private key *(opt)* ECDH public key *(opt)*	R R R
ECDSA key check	Public Key	✓	✓	ECDSA	✓	EC parameters ECDSA private key *(opt)* ECDSA public key *(opt)*	R R R
ECDSA Sign	Public Key	✓	✓	ECDSA (P-224, P-256, P-384, P-521), SHA-224, SHA-256, SHA-384, SHA-512	✓	EC parameters ECDSA private key	R R
ECDSA Verify	Public Key	✓	✓	ECDSA (P-192, P-224, P-256, P-384, P-521), SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	✓	EC parameters ECDSA public key	R R

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
RSA Sign	Public Key	✓	✓	RSA-PSS (n=2048, 3072), SHA-224, SHA-256, SHA-384, SHA-512	✓	RSA-PSS private key	R
RSA Verify	Public Key	✓	✓	RSA-PSS (n=1024 to 3072), SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	✓	RSA-PSS public key	R
RSA Sign	Public Key	✓	✓	RSA-PKCS#1-v1.5 (n=2048, 3072), SHA-224, SHA-256, SHA-384, SHA-512	✓	RSA-v1.5 private key	R
RSA Verify	Public Key	✓	✓	RSA-PKCS#1-v1.5 (n=1024 to 3072), SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	✓	RSA-v1.5 public key	R
ECDSA generate public key	Public Key	✓	✓	ECDSA (P-224, P-256, P-384, P-521)	✓	ECDSA private key EC parameters ECDSA public key	R R C
ECDSA generate private and public key	Public Key	✓	✓	ECDSA (P-224, P-256, P-384, P-521), CTR_DRBG	✓	ECDSA private key EC parameters ECDSA public key	C R C
ECDH generate public key	Public Key	✓	✓	ECDH (P-192, P-224, P-256, P-384, P-521)		ECDH private key EC parameters ECDH public key	R R C
ECDH generate private and public key	Public Key	✓	✓	ECDH (P-192, P-224, P-256, P-384, P-521), CTR_DRBG		ECDH private key EC parameters ECDH public key	C R C
ECDH generate shared secrets (single key-pair)	Public Key	✓	✓	ECDH (P-224, P-256, P-384, P-521), CTR_DRBG		ECDH private key EC parameters ECDH public key ECDH shared secret	R R R C
ECDH key agreement (single key-pair)	Public Key	✓	✓	ECDH (P-224, P-256, P-384, P-521), CTR_DRBG, one-step KDF [SP800-56C], section 4.1 (SHA-256)		ECDH private key EC parameters ECDH public key Derived key	R R R C
ECDH generate shared secrets (dual key-pair)	Public Key	✓	✓	ECDH (P-224, P-256, P-384, P-521), CTR_DRBG		ECDH private keys (2x) EC parameters ECDH public keys (2x) ECDH shared secret	R R R C

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
ECDH key agreement (dual key-pair)	Public Key	✓	✓	ECDH (P-224, P-256, P-384, P-521), CTR_DRBG, one-step KDF [SP800-56C], section 4.1 (SHA-256)		ECDH private keys (2x) EC parameters ECDH public keys (2x) Derived key	R R R C
AES Key wrapping	AES (Un)Wrap	✓	✓	AES Key Wrap with or without padding as specified in [SP800-38F]	✓	AES key	R/R*
AES Key unwrapping	AES (Un)Wrap	✓	✓	AES Key Wrap with or without padding as specified in [SP800-38F]	✓	AES key	R/R*
TRNG Configuration	TRNG Configuration		✓		✓	n/a	
Get TRNG Random Number	TRNG Get Random Number	✓	✓	TRNG	✓	Raw random data to seed DRBG	C*
Get DRBG Random Number	TRNG Get Random Number	✓	✓	CTR_DRBG	✓	Seed Internal DRBG state Internal Counter	RU RU C*
TRNG Post-processing Verification	TRNG Post-Processing Verification	✓	✓		✓	n/a	
TRNG Hardware Self-test Verification	TRNG Hardware Self-test Verification	✓	✓		✓	n/a	
Dynamic Asset Creation	Asset Create	✓	✓		✓	Asset	C
Static Asset Search	Static Asset Search	✓	✓		✓	n/a	
Dynamic Asset Key Derivation	Asset Load (derive)	✓	✓	[SP800-108] KDF two-step KDF [SP800-56C], section 5.1 (HMAC-SHA-256 or AES-CMAC)	✓	Asset Key Derivation Key (KDK)	U R
Dynamic Asset Import as keyblob	Asset Load (import)	✓	✓	AES-CMAC, AES-CTR	✓	Asset Key Encryption Key (KEK)	U R
Dynamic Asset Import as AES wrapped keyblob	Asset Load (AES Unwrap)	✓	✓	AES-KW, AES-KWP	✓	Asset AES Wrap Key	U R

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
Dynamic Asset Generation	Asset Load (random)	✓	✓	CTR_DRBG, AES-CMAC, AES-CTR	✓	Asset Key Encryption Key (KEK)[3]	U R
Dynamic Asset Import as plaintext	Asset Load (plaintext)	✓	✓	AES-CMAC, AES-CTR	✓	Asset Key Encryption Key (KEK)[3]	U R
Dynamic Asset Deletion	Asset Delete	✓	✓		✓	Asset	D
Asset Export to Coprocessor	Asset Export via the Coprocessor Interface	✓	✓		✓	Asset	D
Public Data Read	Public Data Read	✓	✓		✓	Asset	R*
Monotonic Counter Read	Monotonic Counter Read	✓	✓		✓	Asset	R*
Monotonic Counter Increment	Monotonic Counter Increment	✓	✓		✓	Asset	RU
OTP Data Write	One-Time-Programmable Data Write		✓		✓	Asset Provisioning Key	CU R
Generate Random HUK	Provision Random HUK		✓	DRBG	✓	Trusted Root Key (HUK) Crypto Officer Identity	C C
System Information (version, status)	System Information	✓	✓		✓	n/a	
Self-Tests (KAT execution)	Self-Test	✓	✓	All relevant FIPS algorithms	✓	All assets in Asset Store	D
On Demand Self-Tests	Hard reset		✓	All relevant FIPS algorithms	✓	All assets in Asset Store	D
User login Initialization	Enter FIPS mode		✓		✓	All assets in Asset Store	D
Module Reset	Reset	✓	✓		✓	All assets in Asset Store	D

[3] The Key Encryption Key (KEK) is required when the asset needs to be exported as a keyblob.

Service	Input Token	Roles		Cryptographic Algorithms	FIPS	Keys and CSPs	Access
		User	CO				
Authenticated Unlock Start	Authenticated Unlock Start	✓	✓	DRBG	✓	ECDSA Authentication Key Authentication State	R U
Authenticated Unlock Verify	Authenticated Unlock Verify	✓	✓	ECDSA P-256	✓	ECDSA Authentication Key Authentication State	R RU
Activate / Deactivate Debug port signals	Set Secure Debug	✓	✓		✓	ECDSA Authentication Key Authentication State	R R
Register Read	Register Read	✓	✓		✓	n/a	
Register Write	Register Write		✓		✓	n/a	
Clock Switch	Clock Switch		✓			n/a	
Zeroize Output Mailbox	Zeroize Output Mailbox	✓	✓		✓	CSPs in output mailbox	D
Create / Update User Identity	Define Users		✓		✓	User Identity	CU
Select OTP Zeroize	Select One-Time-Programmable Zeroize		✓		✓	n/a	
Zeroize OTP	Zeroize One-Time-Programmable		✓		✓	CSPs in the OTP and Asset Store	D
Go to Sleep mode	Sleep Mode	✓	✓		✓	n/a	
Resume from Sleep mode	Resume from Sleep	✓	✓		✓	n/a	
Set the System timer	Set System Time		✓		✓	n/a	

Table 13 - VaultIP Services

The following table shows the FIPS-Approved algorithms, which can be used in services in FIPS mode of operation, with their corresponding CAVP certificates.

Note: Not all of the algorithms/modes verified through the CAVP certificates are available from the module.

Algorithm	Usage	Key lengths	Modes	Standards	CAVP certs
AES	Encryption Decryption	128, 192, 256 bits	ECB, CBC, CTR	[FIPS197] [SP800-38A]	#C1224
	Encryption Decryption	128, 192, 256 bits	CCM	[SP800-38C]	#C1224
	Encryption Decryption	128, 256 bits	XTS	[SP800-38E]	#C1224
	Encryption Decryption	128, 192, 256 bits	GCM	[SP800-38D]	#C1224 #C1242
	MAC	128, 192, 256 bits	CMAC, GMAC	[SP800-38B] [SP800-38D]	#C1224
AES Key Wrapping (KTS)	Key Wrapping	128, 192, 256 bits	KW, KWP	[SP800-38F] [RFC3394] [RFC5649]	#C1224
AES Key Wrapping with AES-CMAC and AES-CTR (KTS)	Key Wrapping	256-bit AES keys	AES-CMAC and AES-CTR	[SP800-38B] [SP800-38A]	#C1224
DRBG	Random Number Generation		AES-256 in CTR mode, no derivation function, prediction resistance disabled	[SP800-90A] [SP800-38A]	#C1224
ECDSA	Key Verification Signature Verification	P-192, P-224, P-256, P-384, P-521		[FIPS186-4]	#C1224
	Integrity test of Firmware RAM (Signature Verification)	P-256		[FIPS186-4]	#C1224
	Key Generation Signature Generation	P-224, P-256, P-384, P-521		[FIPS186-4]	#C1224
HMAC-SHA-1	MAC Verification	112-512 bits		[FIPS198-1]	#C1224
HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	MAC Generation MAC Verification	112-512 bits 128-512 bits 128-512 bits 128-512 bits		[FIPS198-1]	#C1224
SP800-108 KDF	Key Derivation	128-512 bits	Counter and feedback modes using CMAC-AES-256 or HMAC-SHA-256	[SP800-108] [FIPS198-1] [SP800-38B]	#C1224
SP800-56C KDF (KDA, vendor affirmed)	Key Derivation	128-512 bits	two-step KDF, Counter and feedback modes using CMAC-AES-256 or HMAC-SHA-256 per [SP800-56C], section 5.1	[SP800-56C] [SP800-108] [FIPS198-1] [SP800-38B]	N/A

RSA	Signature Verification	n=(1024 to 3072)	RSA-PSS (no CRT)	[PKCS#1] [FIPS186-4]	#C1224[4]
	Signature Generation	n=(2048 to 3072)	RSA-PSS (no CRT)	[PKCS#1] [FIPS186-4]	#C1224
	Signature Verification	n=(1024 to 3072)	RSA-PKCS#1v1.5 (no CRT)	[PKCS#1] [FIPS186-4]	#C1224
	Signature Generation	n=(2048 to 3072)	RSA-PKCS#1v1.5 (no CRT)	[PKCS#1] [FIPS186-4]	#C1224
SHA-1[5]	Message Digest			[FIPS180-4]	#C1224
SHA-224, SHA-256, SHA-384, SHA-512	Message Digest			[FIPS180-4]	#C1224
TRNG	Random Number Generation			[SP800-90B]	ENT

Table 14 - FIPS approved cryptographic algorithms

The following table shows the cryptographic algorithms and their key lengths that are not FIPS-Approved. These algorithms can only be used in non-FIPS mode of operation.

Algorithm	Usage	Key lengths	Modes	Standards
AES	Encryption Decryption	128, 192, 256 bits	f8	
AES-CBC-MAC	MAC	128, 192, 256 bits	CBC	
DES	Encryption Decryption	56 bits	ECB, CBC	[FIPS46-3]
ECDSA	Key Generation and Signature Generation	P-192		[FIPS186-4] [SP800-131A]
ECDH	Key Generation and Key Verification	P-192, P-224, P-256, P-384, P-521		[FIPS186-4] [SP800-131A]
ECDH	Shared Secret Computation, Key Agreement	Curve25519		
ECDH	Shared Secret Computation	P-192, P-224, P-256, P-384, P-521		
ECDH	Key Agreement	P-192, P-224, P-256, P-384, P-521	ECDH shared secret and one-step KDF using SHA-256 per [SP800-56C], section 4.1	[SP800-56C]

[4] This certificate includes an entry for RSA signature verification compliant with FIPS 186-2, but this algorithm is actually not used by the module.
[5] SHA-1 can be used in FIPS-approved mode for message digest, MAC verification and signature verification algorithms.

edDSA	Key Generation, Signature Generation and Verification	Ed25519		
HMAC-SHA-1	MAC	< 112 bits		[FIPS198-1] [SP800-131A]
RSA	Signature Generation	1024, 1536, and 4096 bits or more	RSA-PSS (no CRT) RSA-PKCS#1-v1.5 (no CRT)	[PKCS#1] [SP800-131A]
	Signature Verification	4096 bits or more		
SHA-1	Signature Generation			[FIPS180-4] [SP800-131A]
Triple-DES	Encryption Decryption	192 bits	ECB, CBC	[SP800-67], [SP800-38A]

Table 15 - Non-FIPS approved cryptographic algorithms

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

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

Table 16 - 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.

Note: Integrity of the RAM Firmware is performed using ECDSA signature verification at FW load time. When resuming from sleep mode, RAM Firmware integrity is verified by SHA-256.

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 and Crypto Officer roles for each service request; role identification and authentication status are not internally maintained in the cryptographic module.

The module is initialized via the "*Provision Random HUK*" token, which instructs the module to generate the Trusted Root key (HUK) and assigns the Crypto Officer 32-bit ID. The module supports only one Crypto Officer role identity; this value is stored in One Time Programmable (OTP) memory and cannot be altered unless the whole module is zeroized.

To access the module to request services, 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.

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.

In order to initialize the User Login, the CO first needs to call *Define User* service to set up the Users and then call the *User Login Initialization* service. The Crypto Officer role is a default role that is configured during module install and therefore no specific Login initialization service is required here.

4.4 Mechanism and Strength of Authentication

Because there is a single 32-bit Crypto Officer Identity, the probability of successfully guessing the Crypto Officer Identity is $1/2^{32}$. Similarly, the probability of successfully guessing one of up to four 32-bit User Identities 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 Xilinx Zynq XC7Z045 FPGA. This FPGA is a single chip that includes standard passivation provided by an integrated heat spreader (IHS). The IHS serves as a protective shell around the processing silicon, a pathway for heat to be exchanged between the SoC and SoC cooler, providing opacity in the visible spectrum and preventing any access to the interior of the chip. The FPGA 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 assets:

- *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 7.6).
- *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 Data RAM 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 17 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 7.3.2 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 7.3.1 for more information.

Name	Key Length[6] / CSP Size	Generation			Entry				Storage		Output
		KBKDF	Random	Key Pair	Firmware	Plaintext	KeyWrap	KeyBlob	Static	Dynamic	
Trusted Root Key (HUK)	128, 256 bits		✓					✓	✓		
Trusted Key Encryption Keys (KEK)	256 bits	✓								✓	
Trusted Key Derivation Keys (KDK)	128, 256 bits	✓							✓	✓	
Authentication Keys (ECDSA public key with P-256 curve)	256 bits				✓				✓		
Provisioning Key	2x256 bits				✓				✓		
AES keys	128, 192, 256 bits	✓	✓			✓	✓	✓		✓	✓[7]
HMAC keys	SHA-1: 112-512 bits SHA-224: 112-512 bits SHA-256: 128-512 bits SHA-384: 1024 bits SHA-512: 1024 bits	✓	✓			✓	✓	✓		✓	✓[7]
RSA public and private keys	1024-3072 bits					✓	✓	✓		✓	✓[8]
ECDSA public and private keys	192-521 bits			✓		✓	✓	✓		✓	✓[9]
Crypto Officer Identity	32 bits					✓			✓		
User Identity	32 bits					✓				✓	
Entropy input string										✓	
DRBG internal state and counter										✓	
Authentication state										✓	

Table 17 - Life Cycle of Keys and Critical Security Parameters (CSPs)

All CSPs that VaultIP processes are listed in the “Keys and CSPs” column in Table 13, together with their required access (create, read, update, delete). They reside in the internal memory of VaultIP and get zeroized when the module is powered off, with the exception of CSPs stored in OTP.

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 keys that are located in firmware, the Authentication Keys, are public keys.

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

7.1 Key Generation

VaultIP provides services for generating asymmetric and symmetric keys. The key generation methods implemented in the module are compliant with [SP800-133] (vendor affirmed).

VaultIP implements symmetric key generation for the Key Encryption Key (KEK), and AES and HMAC keys, via the *"Asset Load (random)"* token, using random data obtained from a Deterministic Random Bit Generator (DRBG) compliant with [SP800-90A].

VaultIP implements Elliptic Curve DSA (ECDSA) asymmetric key generation services compliant with [FIPS186-4]. A seed (i.e. the random value) used in asymmetric key generation is directly obtained from the [SP800-90A] DRBG.

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

7.2 Key Derivation

VaultIP provides services for deriving keys from the Trusted Root Key (HUK), or any asset indicated as a Trusted Key Derivation Key (KDK). VaultIP provides the following key-based derivation algorithms:

- Key-based key derivation function (KBKDF) in Counter or Feedback modes using HMAC-SHA-256 or AES-CMAC, in compliance with [SP800-108].
- Two-step key derivation function (KDF) using HMAC-SHA-256 or AES-CMAC, in compliance with section 5.1 of [SP800-56C] (vendor affirmed).

***Vendor Affirmation**: key derivation function compliant with [SP800-56C] has not been tested by the CAVP. The CMVP or CAVP provide no assurance regarding its correct implementation or operation. Rambus affirms that the method was implemented correctly.*

7.3 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.

7.3.1 Dynamic assets

Dynamic assets (AES keys, HMAC keys, RSA public and private keys, ECDSA public and private keys) 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.

7.3.2 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 ECDSA Authentication Public Keys generated with NIST curve P-256.
- One 512-bit AES Provisioning Key (AES-CMAC, AES-CTR).

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

7.4 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 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.

7.5 Key Agreement / Key Transport

VaultIP provides key wrapping. As shown in Table 17 and explained in section Dynamic assets, keys can be entered in encrypted form through the following key wrapping methods and key lengths:

- Key Wrapping with AES in KW and KWP modes with 128, 192 or 256-bit keys, compliant with [SP800-38F] and IG D.9. Security strength ranges from 128 to 256 bits, according to the AES key length.
- Key Wrapping using AES in CMAC and CTR modes with two 256-bit keys (the first key for the AES-CMAC operation and the second key for the AES-CTR operation), compliant with [SP800-38F] and IG D.9. It provides security strength of 256 bits.

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 agreement or 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
HMAC key	256 bits	256 bits
RSA key pair	3072 bits	128 bits
ECDSA key pair	521 bits	256 bits

Table 18 - 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.

7.6 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 the *Enter FIPS Mode* token or the *Reset* token are invoked, when VaultIP 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 fills the complete OTP with ones, and zeroizes 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 synchronously and in the microsecond range, which 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 "User login initialization" service is invoked.
- No information is output when VaultIP transitions to or is in the Error state.

7.7 Random Number Generation

VaultIP includes a Deterministic Random Bit Generator (DRBG) based on the CTR_DRBG (without prediction resistance; without derivation function) 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 *ECDSA sign/verify* service.
- Provide random data for the *TRNG Get Random* service.

- Create the internal Initialization Vector (IV) for the AES GCM.

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. The module also performs DRBG health tests according to section 11.3 of SP 800-90A.

7.8 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.

The TRNG meets the requirements of [SP800-90B].

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 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 with the following status signals: 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: fatal_error =1b

9.1.1 Firmware ROM 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 Firmware ROM Cryptographic Algorithm tests

Once the ROM contents have been successfully verified, ROM-based self-tests are automatically executed in preparation for the execution of the boot process. The executed tests are shown in the following table.

Cryptographic Algorithm	Test
SHS	KAT SHA-256
ECDSA	KAT for ECDSA (NIST P-256) signature verification
AES	KAT AES-CTR, 256-bit, decrypt

Table 19 ROM 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.

9.1.3 Firmware RAM Integrity tests

VaultIP verifies the integrity of the firmware image received by the host using ECDSA signature verification with a P-256 public key. If verification of the ECDSA digital signature succeeds, VaultIP continues with the rest of the power-up tests; if the integrity test fails, the module does not proceed with the power-up and waits for a valid firmware image to be loaded.

9.1.4 Firmware RAM 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[10]	Test
AES	KAT AES-CBC, 128-bit, encryption KAT AES-CBC, 128-bit, decryption KAT AES-CCM, 192-bit, encryption KAT AES-CCM, 192-bit, decryption KAT XTS-AES, 256-bit, encryption KAT XTS-AES, 256-bit, decryption KAT AES-GCM, 256-bit, encryption KAT AES-GCM, 256-bit, decryption KAT AES-CMAC, 256-bit, MAC generation
SHS	KAT SHA-1 KAT SHA-224 KAT SHA-384 KAT SHA-512
HMAC	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
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
KBKDF	KAT SP800-108 KDF
CRC32	KAT CRC32

Table 20 - 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. However, this service does not perform the same tests executed during power-up tests; the integrity tests on the firmware ROM and RAM are not executed.

In order to perform the on-demand self-tests as required by FIPS 140-2, the Crypto-Officer shall power-off and power-on VaultIP by doing a hard reset.

9.3 Conditional Tests

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

[10] The module also includes KAT for ECDH shared secret computation but it is a non-approved algorithm hence it is not listed in this table.

Algorithm	Test
DRBG	Continuous test.
ECDSA key generation	Pair-wise consistency test.
TRNG	Adaptive Proportion Test and Repetition Count Test per [SP800-90B] health tests.

Table 21 - Conditional Tests

If any of these tests fail, VaultIP transitions to the Error state and becomes non-operational with the following status signals: fatal_error =1b

9.4 Module status

VaultIP provides different interfaces to show its status:

- The *fatal_error* output signal (equivalent to bit 31 of the MODULE_STATUS register accessible by hosts) indicates that the cryptographic module is in an Error state. This information is available to any user.
- The *System Info* service provides the 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 when resuming from Sleep.
- 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, only the *System Info* and *Reset* services are available. No other services are available and data output is inhibited.

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

- Hardware reset the module (System Abort Request).
- Power-off and power-on the module.

10 Design Assurance

10.1 Configuration Management

Rambus uses a configuration management system to store all source code, test tools and verification scripts. A change and version control tool is used to identify and track each configuration item. 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:

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

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

where:

- <nn> is the “EIP number” in Rambus’s IP catalog.
- <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.

10.1.2 Guidance Identification

Rambus uniquely identifies each document with the document number (e.g. 007-130300-201) and Revision (RevA, RevB, etc.).

10.1.3 Source Code Identification

Source code is identified and controlled by the configuration management tool. Proper keywords are included in each source code file to provide the filename and revision configuration item.

10.2 Delivery and Operation

VaultIP synthesized in the Xillinx Zynq XC7Z045 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:

- 915-130031-300_EIP130-100_HW3.0.3.zip
- 914-130031-300_EIP130-100_Firmware-cfgA_FW3.0.6.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 Xillinx Zynq XC7Z045 FPGA and validated as a single-chip hardware module. Nevertheless, VaultIP, as a soft IP core written in RTL code, can also be synthesized in other single-chip implementations with different chip technologies and/or different non-security relevant functional subsystems. FIPS 140-2 IG 1.20 in [FIPS140-2_IG] provides guidance on how to take advantage of a sub-chip validated module and re-validate the module in other single-chip implementations.

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

Document Name	Document Number	Revision
VaultIP-130 Crypto Officer and User Manual	007-130300-403	RevB
VaultIP-130 Security Module, Integration Manual	007-130300-200	RevC
VaultIP-130 Security Module, Hardware Reference Manual	007-130300-201	RevD
VaultIP-130 Security Module, Firmware Reference Manual	007-130300-204	RevA

Table 22 - VaultIP Documentation

The following sections provide further guidance for algorithms in FIPS mode of operation.

10.3.1 TRNG Configuration

The TRNG must be correctly configured to ensure that sufficient entropy is generated and sufficient bits are provided to the conditioning function. The validated module running on the Xillinx Zynq XC7Z045 FPGA has been tuned to comply with SP800-90B and ensure one bit of entropy per bit. Therefore, VaultIP shall be configured to work with the TRNG configuration parameters as described in section 4.9 of the “VaultIP-130 Crypto Officer and User Manual”. These parameters should be used in the following services:

- TRNG Configuration
- Provision Random HUK

For synthesizing VaultIP in other chips or systems, the TRNG configuration parameters shall need to be adjusted in order to provide the same entropy rate.

10.3.2 AES GCM IV

In case the module’s power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

VaultIP is compliant with scenario 2 of FIPS 140-2 IG A.5 in [FIPS140-2_IG]. The internal IV is generated in the encryption operation using the RBG-based construction method as defined in section 8.2.2 of [SP800-38D]. VaultIP generates a IV with a length of 128 bits; the first 96 bits are initialized with random data obtained from the SP800-90A DRBG implemented in the module.

10.3.3 AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E].

VaultIP implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical, meeting the requirement of FIPS 140-2 IG A.9 in [FIPS140-2_IG].

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
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
GCM	Galois/Counter Mode
HMAC	Hash Message Authentication Code
ICM	Integer Counter Mode
HIS	Integrated Heat Spreader
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
TEE	Trusted Execution Environment
UI	User Interface

A.2 References

FIPS140-2 **FIPS PUB 140-2 - Security Requirements For Cryptographic Modules**
May 2001
https://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
August 28, 2020
https://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed-Hash Message Authentication Code (HMAC)**
July 2008
https://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
https://www.ietf.org/rfc/rfc3447.txt

RFC3394 **Advanced Encryption Standard (AES) Key Wrap Algorithm**
September 2002
https://www.ietf.org/rfc/rfc3394.txt

RFC5649 **Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm**
September 2009
https://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
https://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
https://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
https://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf

SP800-38D **NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**
November 2007
https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.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
https://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
https://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
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf

SP800-56C **NIST Special Publication 800-56C Revision 1 - Recommendation for Key-DerivationMethods in Key-Establishment Schemes**
April 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf

SP800-67 **NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
January 2012
https://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

SP800-90A **NIST Special Publication 800-90A – Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
January 2015
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-90B **NIST Draft Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation**
January 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

SP800-108 **NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions**
October 2009
https://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf