Security Target

for LANCOM Business VPN Router

'LANCOM 1900EF'

with LANCOM Systems Operating System

'LCOS 10.32.0029 PR'

and IPsec VPN

Version 1.26

Release

Document History

Date	Version	Editor	Changes
14.04.2019	1.0	FTheinen	First Document Release
21.05.2019	1.10	FTheinen	Added Cryptographic Specification & Key Management Description
13.09.2019	1.11	FTheinen	Editorial Change: 'P1010 SEC' -> 'QorIQ SEC'
13.09.2019	1.20	FTheinen	Replaced TOE: 'LANCOM 1900EF', 'LCOS 10.32 Rel' Updated sections 'Acronyms', 'General Description', 'Interfaces', 'Security Functions', 'Limits of Evaluation' Updated appendices 'Architecture Overview', 'Update Mechanism', 'Random Source Description' Replaced SSH authentication algorithm: 'rsa-ssh' -> 'ssh-sha2-256, ssh-sha2-512' (RFC 8332)
21.01.2021	1.25	FTheinen	Updated TOE version to 'LCOS 10.32.0029 PR' Updated Security Function SecFunc.Firewall.DoS.IDS
08.06.2021	1.26	FTheinen	Removed "RSA signature generation and verification (RSASSA-PKCS1-v1_5) using SHA-1" from Cryptographic Specification

Table of Contents

Introduction

Context of This Document

This document is the Security Target (ST) for the BSZ certification of the "LANCOM Business VPN Router 'LANCOM 1900EF' with LANCOM Systems Operating System 'LCOS 10.32.0029 PR' and IPsec VPN" (Target of Evaluation, TOE) at the BSI. The configuration chosen is the typical use case of the TOE.

The document itself was written by Frank Theinen (Embedded Systems Engineer, LANCOM Systems GmbH). Review was performed by Thomas Jansen (Embedded Systems Engineer, LANCOM Systems GmbH). It was released by Stefan Guddat (Product Manager, LANCOM Systems GmbH).

Note: This document uses gender-neutral language, especially the 'singular they' instead of the 'generic he'.

Product Identification

TOE name: 'LANCOM 1900EF'

TOE version: 'LCOS 10.32.0029 PR' -> '10.32.0029PR / 21.01.2021'

The TOE name and version are displayed on the system information page in the web-based management interface (WEBconfig). Depending on the configured web browser language (EN/DE), they are shown under:

- 'System information' > 'System data' > 'Device type'
- 'System information' > 'System data' > 'Firmware version'

or

- 'Systeminformation' > 'Systemdaten' > 'Gerätetyp'
- 'Systeminformation' > 'Systemdaten' > 'Firmwareversion'

The TOE name and version are also displayed in the banner of the command line interface (CLI). The TOE name is additionally printed at two locations on the physical device: on the front side and on the bottom side label.

References / Acronyms

Acronyms	
BSI	Bundesamt für Sicherheit in der Informationstechnik
BSZ	Beschleunigte Sicherheitszertifizierung
CLI	Command Line Interface
COM	Communication
DoS	Denial-of-service
DSL	Digital Subscriber Line
HTTP	Hypertext Transfer Protocol

Acronyms	
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection/Prevention Services
IP	Internet Protocol
IPsec	IP security
IT	Information Technology
LAN	Local Area Network
LCOS	LANCOM Systems Operating System
MITM	Man-in-the-middle
NAT	Network Address Translation
RFC	Request for Comments (IETF Standard)
SCP	Secure Copy
SFP	Small Form-factor Pluggable
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
ST	Security Target
SUG	Secure User Guidance
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOE	Target of Evaluation
TP	Twisted Pair
UDP	User Datagram Protocol
USB	Universal Serial Bus
VPN	Virtual Private Network
WAN	Wide Area Network
WEBconfig	Web-based management interface

References		
RFC 3411	An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks	https://tools.ietf.org/html/rfc3411
RFC 4251	The Secure Shell (SSH) Protocol Architecture	https://tools.ietf.org/html/rfc4251
RFC 4301	Security Architecture for the Internet Protocol	https://tools.ietf.org/html/rfc4301
RFC 5246	The Transport Layer Security (TLS) Protocol Version 1.2	https://tools.ietf.org/html/rfc5246

References		
RFC 7230	Hypertext Transfer Protocol (HTTP/1.1)	https://tools.ietf.org/html/rfc7230

Product Description

General Description

The LANCOM Business VPN Router 'LANCOM 1900EF' offers secure VPN site connectivity over high-performance Internet connections. It has two Gigabit Ethernet WAN ports and works with high-speed fiber-optic connections and any external DSL or cable modems. A stateful inspection firewall protects the whole network with features such as intrusion prevention and Denial-of-service protection.

Features

Roles

The TOE supports two independent roles: an administrator and a user:

- The administrator installs and manages the TOE. They have physical access to the TOE for the physical installation (e.g. cabling). They use trusted network access to configure and monitor the TOE and to update the TOE firmware.
- The user communicates through the TOE. They have no physical access to the TOE. If allowed by the TOE's configuration, they use the Internet access, IPsec VPN, firewall and routing services of the TOE.

Interfaces

The TOE has 4 LAN Ethernet ports, 2 WAN Ethernet ports, a COM (serial) port and an USB port:

- The LAN Ethernet ports support Gigabit Ethernet (1 Gbit/s). They can be used individually to connect the TOE to either trusted local networks (e.g. LAN) or untrusted remote networks (e.g. Internet).
- The WAN Ethernet ports support Gigabit Ethernet (1 Gbit/s). They can be used individually to connect the TOE to either trusted local networks (e.g. LAN) or untrusted remote networks (e.g. Internet). Note that one of the WAN Ethernet ports is a SFP/TP combo port, and therefore has two sockets.
- The COM (serial) port supports serial data communication (up to 115 kbit/s). It can be used to directly connect a computer to the TOE for management purposes.
- The USB port supports USB 2.0 (480 Mbit/s). It can be used for management purposes.

The typical configuration of the TOE consists of trusted local networks connected to LAN Ethernet ports and untrusted remote networks connected to WAN Ethernet ports. Management of the TOE is done using only trusted networks, and the COM and USB ports are not used. More complex configurations are possible but out of scope for this ST.

Management Services

The administrator can manage the TOE using the following interfaces:

- WEBconfig: An HTTPS server provides a graphical user interface to configure and monitor the TOE and to update the TOE firmware using a web browser.
- CLI: An SSH server provides a command line interface to configure and monitor the TOE using an SSH client. Updating the TOE firmware can be done using Secure Copy (SCP).
- SNMP: An SNMPv3 server allows SNMP monitoring software to monitor the TOE.

Internet Access

The TOE provides the user in trusted local networks (e.g. LANs) with access to untrusted remote networks (e.g. Internet). If the access to untrusted remote networks uses IPv4 Network Address Translation (NAT),

- communication sessions originating from trusted local networks are automatically allowed, whereas
- communication sessions originating from untrusted remote networks are automatically denied.

More detailed control over allowing and denying communication sessions can be configured by the administrator in the firewall.

IPsec VPN Connections

The TOE provides IPsec VPN services for the user. IPsec VPN provides confidentiality, integrity and authenticity for user data transmitted over untrusted remote networks (e.g. Internet) by encrypting and authenticating the user data. It can be used

- to connect trusted local networks (e.g. LANs in local company branch) with trusted remote networks (e.g. LANs in remote company branches) over untrusted remote networks (e.g. Internet) and/or
- to connect trusted mobile devices (e.g. road warriors) with trusted local networks (e.g. LANs in local company branch) over untrusted remote networks (e.g. Internet).

IPv4 Firewall / Routing

The TOE provides IPv4 firewall and routing services for the user. The IPv4 stateful packet inspection firewall allows or denies communication sessions according to its configuration by the administrator. Additionally it provides Denial-of-service (DoS) protection and intrusion detection/prevention services (IDS). When a communication session is allowed by the firewall, the path to the destination is determined by the routing service.

The TOE additionally provides IPv6 protocol services that are out of scope for this ST.

Product Usage

General Concepts

The TOE is intended as an edge router to separate trusted local networks (e.g. LANs) from untrusted remote networks (e.g. Internet). The IPv4 stateful packet inspection firewall of the TOE allows or denies communication sessions to/from/over the untrusted remote networks (e.g. Internet) according to its configuration. In particular, it can allow:

- Internet access: Users in trusted local networks (e.g. LANs) are provided with access to untrusted remote networks (e.g. Internet).
- IPsec VPN connections: Trusted local networks (e.g. LANs in local company branch) can be connected to trusted remote networks (e.g. LANs in remote company branches) over untrusted remote networks (e.g. Internet) using IPsec VPN, which provides confidentiality, integrity and authenticity by encrypting and authenticating the user data transmitted.
- IPsec VPN connections: Trusted mobile devices (e.g. road warriors) can be connected to trusted local networks (e.g. LANs in local company branch) over untrusted remote networks (e.g. Internet) using IPsec VPN, which provides confidentiality, integrity and authenticity by encrypting and authenticating the user data transmitted.

By the Administrator

The administrator of the TOE uses the management interfaces WEBconfig and/or CLI to configure:

- the trusted local networks (e.g. LANs),
- the untrusted remote networks (e.g. Internet),
- the IPsec VPN connections,
- the IPv4 firewall and routing services and
- other security relevant settings.

The administrator also uses the management interfaces to monitor the TOE (WEBconfig, CLI and/or SNMP) and to update the TOE firmware (WEBconfig and/or CLI).

Details can be found in the Secure User Guidance (SUG) document.

By the User

If allowed by the TOE's configuration, the user communicates through the TOE

- from within trusted local networks (e.g. LANs) with services in the untrusted remote networks (e.g. Internet),
- from within trusted local networks (e.g. LANs in local company branch) with trusted remote networks (e.g. LANs in remote company branches) over untrusted remote networks (e.g. Internet) using IPsec VPN connections and/or
- from trusted mobile devices (e.g. road warriors) with trusted local networks (e.g. LANs in local company branch) over untrusted remote networks (e.g. Internet) using IPsec VPN connections.

Operating Environment

The TOE is intended for environments with physical access restrictions and trusted local users. Remote users are either trusted or untrusted.

The administrator has physical access to the TOE for the physical installation (e.g. cabling). They can manage the TOE using the following protocols over IPv4:

- WEBconfig: HTTP/1.1 [RFC 7230 ff.] over TLS 1.2 [RFC 5246] (HTTPS)
- CLI: SSH [RFC 4251 ff.]
- SNMP: SNMPv3 [RFC 3411 ff.]

The user has no physical access to the TOE. If allowed by the TOE's configuration, they communicate through the TOE using the IPv4 protocol:

- Users in trusted local networks (e.g. LANs) are trusted.
- Users in untrusted remote networks (e.g. Internet) are untrusted in general, but are trusted if allowed by the TOE's configuration.
- Users in trusted remote networks (e.g. LANs in remote company branches) connected using IPsec VPN are trusted.
- Users with trusted mobile devices (e.g. road warriors) connected using IPsec VPN are trusted.

The IPsec VPN service uses the following protocol:

- IPsec [RFC 4301 ff.]

The TOE additionally provides IPv6 protocol services that are out of scope for this ST.

Security Perimeter

Users

For the TOE, the following users exist:

- Administrator: Mapped to the administrator role. The administrator installs and manages the TOE. They have physical access to the TOE for the physical installation (e.g. cabling). They use the WEBconfig, CLI and SNMP management interfaces to configure and monitor the TOE and to update the TOE firmware. They are allowed to establish connections to the management interfaces either from trusted local networks (e.g. LANs), or from trusted remote networks (e.g. LANs in remote company branches) over untrusted remote networks (e.g. Internet) using IPsec VPN connections to the TOE. They are authenticated either via username and password or via an SSH key.
- Normal User: Mapped to the user role. This user has no physical access to the TOE, and is not allowed to use the management interfaces of the TOE. They communicate through the TOE, if allowed by the TOE's configuration, by using the Internet access, IPsec VPN, firewall and routing services of the TOE. They are identified by the source IP address and, depending on the IP protocol (e.g. TCP, UDP), the TCP or UDP source port of the first IP packet of a communication session.

Note: The notion 'normal user' is only used in this section to differentiate it from this section's topic 'Users'. In all further sections, they are simply called 'user' again.

Assumptions

For the TOE to fulfill its security properties, the following assumptions must apply:

- **Assumption.OnlyConn** - The TOE shall be the only logical and physical connection between the trusted local networks (e.g. LANs) and the untrusted remote networks (e.g. Internet). Otherwise, the TOE cannot keep the networks separated on its own.
- **Assumption.PhysAcc** - The physical access to the TOE shall be limited to trustworthy personnel. Otherwise, an attacker could perform physical attacks against the TOE (e.g. attaching a hardware debugger to read or change the TOE's configuration). Additionally an attacker could physically connect a MITM device to trusted local networks (e.g. LANs) right next to the TOE (e.g. to perform sniffing or DNS redirecting attacks).
- **Assumption.AdminNoEvil** - The administrator of the TOE shall be trustworthy personnel. Otherwise, the administrator could deliberately configure the TOE to not fulfill its security properties, e.g. allow users from untrusted remote networks (e.g. Internet) to access trusted local networks (e.g. LANs).
- **Assumption.AdminKnowHow** - The administrator of the TOE shall be able to configure the TOE securely. Otherwise, the administrator could unknowingly configure the TOE to not fulfill its security properties, e.g. allow users from untrusted remote networks (e.g. Internet) to access trusted local networks (e.g. LANs).
- **Assumption.AdminSecCreds** - The administrator of the TOE shall be able to securely generate administrator credentials (e.g. password, SSH key) and IPsec VPN credentials (e.g. pre-shared keys, RSA keys and certificates). Otherwise, the created credentials may be not strong enough to withstand an attacker.

- **Assumption.AdminSecComp** - The administrator of the TOE shall use a secure computer for the management of the TOE. Otherwise, an attacker could attack the administrator's computer, e.g. to install a keylogger to get access to the administrator credentials or to the TOE's configuration.
- **Assumption.AdminSecAssets** - The administrator of the TOE shall put copies of the TOE's configuration and other valuable assets of the TOE in a secure place when storing them outside of the TOE. Otherwise, an attacker could try to read or change a copy of the TOE's configuration outside of the TOE.
- **Assumption.IPsecPeersTrusted** - The administrator shall configure the TOE to establish IPsec VPN connections only with other trusted IPsec VPN peers (e.g. another copy of the TOE used according to this ST and the SUG). Otherwise, the TOE cannot keep the IPsec VPN connections secured on its own and therefore cannot keep the trusted local networks (e.g. LANs) separated on its own.

Assets

The TOE protects the following assets and security properties thereof:

- **Asset.TOE.Config** - TOE configuration; confidentiality, integrity, authenticity. The TOE configuration consists of the settings of the TOE, which are the TOE's default settings modified and expanded by the administrator (according to the Secure User Guidance). The TOE configuration determines (together with the TOE firmware) how the TOE fulfills its security properties. It is protected inside the TOE as well as outside of the TOE during transmission to/from the TOE. Confidentiality prevents an attacker from gaining knowledge about the TOE configuration, while integrity and authenticity prevent an attacker from changing the TOE configuration.
- **Asset.TOE.MonData** - TOE monitoring data; confidentiality, integrity, authenticity. The TOE monitoring data consists of information about (previous and current) states and events in the TOE that the TOE automatically collects and stores at runtime. Additionally the administrator can command the TOE (in the CLI management interface) to collect tracing information. The TOE monitoring data is protected inside the TOE as well as outside of the TOE during transmission from the TOE. Confidentiality prevents an attacker from gaining knowledge about the TOE monitoring data, while integrity and authenticity prevent an attacker from changing the TOE monitoring data.
- **Asset.TOE.Firmware** - TOE firmware; integrity, authenticity. The TOE firmware contains the operating system of the TOE (LCOS) and determines (together with the TOE configuration) how the TOE fulfills its security properties. It is protected inside the TOE as well as outside of the TOE during transmission to the TOE. Additionally it is protected on the websites it can be downloaded from. Integrity and authenticity prevent an attacker from changing the TOE firmware. Confidentiality is provided inside the TOE and during the transmission to the TOE, but not on the websites it can be downloaded from, where the TOE firmware is signed but not encrypted.
- **Asset.User.Data.LAN** - User data; confidentiality, integrity, authenticity. The user stores data (e.g. data containing company confidential information) in IT devices connected to trusted local networks (e.g. LANs). Additionally the user transmits this data over the trusted local networks (e.g. LANs). The TOE protects the user data by limiting access to the trusted local networks (e.g. LANs) from untrusted remote networks (e.g. Internet). Confidentiality, integrity and authenticity are provided automatically when the access is prevented. Confidentiality prevents an attacker

from gaining knowledge about the user data, while integrity and authenticity prevent an attacker from changing the user data.

- **Asset.User.Data.Inet** - User data; confidentiality, integrity, authenticity. The user transmits data (e.g. data containing company confidential information) between trusted local networks (e.g. LANs in local company branch) and trusted remote networks (e.g. LANs in remote company branches) or trusted mobile devices (e.g. road warriors). The TOE transmits this user data over untrusted remote networks (e.g. Internet) protected by IPsec VPN connections. Confidentiality prevents an attacker from gaining knowledge about the user data, while integrity and authenticity prevent an attacker from changing the user data.

Note: The administrator credentials (e.g. password, SSH key) are part of the TOE configuration and therefore no separate asset.

Note: The IPsec VPN credentials (e.g. pre-shared keys, RSA keys and certificates) are part of the TOE configuration and therefore no separate asset.

Threat Model: Attackers

The following attackers of the TOE are assumed in the threat model:

- **Attacker.Inet** - User in untrusted networks (e.g. Internet) who wants to read or change any of the assets
- **Attacker.LAN** - User in trusted networks (e.g. LANs) who does not have the administrator role and wants to read or change any of the assets only the administrator role is allowed to read or change

Threat Model: Threats

The following threats are expected:

- **Threat.WEBconfig.Access** - The WEBconfig management interface is used by the attacker to read or change the TOE configuration, the TOE monitoring data or the TOE firmware.
- **Threat.WEBconfig.MITM** - The WEBconfig management interface is used by the administrator to read or change the TOE configuration, the TOE monitoring data or the TOE firmware. The attacker reads or changes the transmitted information as a MITM.
- **Threat.CLI.Access** - The CLI management interface is used by the attacker to read or change the TOE configuration, the TOE monitoring data or the TOE firmware.
- **Threat.CLI.MITM** - The CLI management interface is used by the administrator to read or change the TOE configuration, the TOE monitoring data or the TOE firmware. The attacker reads or changes the transmitted information as a MITM.
- **Threat.SNMP.Access** - The SNMP management interface is used by the attacker to read or change the TOE monitoring data.
- **Threat.SNMP.MITM** - The SNMP management interface is used by the administrator to read or change the TOE monitoring data. The attacker reads or changes the transmitted information as a MITM.

- **Threat.LAN.Access** - The user data is read or changed by the attacker accessing the trusted local networks (e.g. LANs) from untrusted remote networks (e.g. Internet).
- **Threat.IPsec.Access** - The user data is read or changed by the attacker accessing the trusted local networks (e.g. LANs) over untrusted remote networks (e.g. Internet) after establishing IPsec VPN connections with the TOE.
- **Threat.IPsec.MITM** - The user data is read or changed by the attacker inside IPsec VPN connections as a MITM.

Security Functions

The following security related functions exist to counter the expected threats:

- **SecFunc.HTTPS** - The TOE implements access to the WEBconfig management interface with HTTP/1.1 [RFC 7230 ff.] over TLS 1.2 [RFC 5246] (HTTPS). The protocols provide the administrator with secure login and secure access to the management interface by providing confidentiality, integrity and authenticity to the administrator credentials, the TOE configuration, the TOE monitoring data and the TOE firmware when being transmitted to/from the TOE.
- **SecFunc.SSH** - The TOE implements access to the CLI management interface with SSH [RFC 4251 ff.]. The protocol provides the administrator with secure login and secure access to the management interface by providing confidentiality, integrity and authenticity to the administrator credentials, the TOE configuration, the TOE monitoring data and the TOE firmware when being transmitted to/from the TOE.
- **SecFunc.SNMPv3** - The TOE implements access to the SNMP management interface with SNMPv3 [RFC 3411 ff.]. The protocol provides the administrator with secure login and secure access to the management interface by providing confidentiality, integrity and authenticity to the administrator credentials and the TOE monitoring data when being transmitted to/from the TOE.
- **SecFunc.IPsec** - The TOE implements IPsec VPN connections with IPsec [RFC 4301 ff.]. The protocol provides the user with secure data transmission over insecure networks by providing confidentiality, integrity and authenticity to the user data when being transmitted over untrusted remote networks (e.g. Internet).
- **SecFunc.IPsec.Log** - The TOE logs successful and unsuccessful IPsec VPN connection establishment attempts within the TOE monitoring data.
- **SecFunc.Firewall.Sessions** - The TOE implements IPv4 firewall and routing services. Using the TOE configuration (according to the SUG), they allow the user in trusted local networks (e.g. LANs) to access untrusted remote networks (e.g. Internet), and they deny the user in untrusted remote networks (e.g. Internet) to access trusted local networks (e.g. LANs). They also allow the user in trusted local networks (e.g. LANs in local company branch) to access trusted remote networks (e.g. LANs in remote company branches) and the user with trusted mobile devices (e.g. road warriors) to access trusted local networks (e.g. LANs in local company branch) using IPsec VPN connections.
- **SecFunc.Firewall.DoS.IDS** - The TOE implements an IPv4 firewall that provides Denial-of-service (DoS) protection and intrusion detection/prevention services (IDS). The DoS protection can detect and react on TCP SYN flooding, Smurf attacks, LAND attacks, Ping of Death attacks, Teardrop attacks, and Bonk attacks. The IDS can detect and react on IP spoofing and port scans.

- **SecFunc.Firewall.Log** - The TOE logs access attempts denied by the firewall within the TOE monitoring data.
- **SecFunc.Auth.AdmCrds** - The TOE authenticates administrators before granting access to the WEBconfig, CLI and SNMP management interfaces, either via username and password or via an SSH key (the latter only being usable in case of the CLI management interface).
- **SecFunc.Auth.AdmPwdChrs** - The TOE enforces the administrator password to contain at least 8 characters from 3 of the following 4 character classes: lowercase letters, uppercase letters, digits and special characters.
- **SecFunc.Auth.BrtFrcCtr** - The TOE counters brute-force attacks against the password by locking the login functionality of an interface for a configured amount of time after a configured number of failed login attempts.
- **SecFunc.Auth.AutoLogOut** - The TOE automatically logs out the administrator after a configured time of inactivity.
- **SecFunc.Auth.Log** - The TOE logs successful and unsuccessful login attempts within the TOE monitoring data.
- **SecFunc.Mgmt.NoInet** - The TOE does not allow access to the WEBconfig, CLI and SNMP management interfaces from untrusted networks in the TOE configuration (according to the SUG).
- **SecFunc.Mgmt.Ports** - The TOE can be configured to use non-standard TCP/UDP ports for the protocols HTTPS, SSH and SNMPv3, which are used to access the WEBconfig, CLI and SNMP management interfaces.

Note: The TOE verifies the integrity and authenticity of the received TOE firmware during a firmware update (according to Appendix 'Update Mechanism').

Threats vs. Assets

The following table shows which security functions protect which assets against which threats by which attackers:

Asset(s)	Attacker(s)	Threat(s)	Security Function(s)
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.Inet	Threat.WEBconfig.Access	SecFunc.Mgmt.NoInet, SecFunc.Mgmt.Ports
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.Inet	Threat.WEBconfig.MITM	SecFunc.IPsec, SecFunc.HTTPS
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.Inet	Threat.CLI.Access	SecFunc.Mgmt.NoInet, SecFunc.Mgmt.Ports
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.Inet	Threat.CLI.MITM	SecFunc.IPsec, SecFunc.SSH

Asset(s)	Attacker(s)	Threat(s)	Security Function(s)
Asset.TOE.MonData	Attacker.Inet	Threat.SNMP.Access	SecFunc.Mgmt.NoInet, SecFunc.Mgmt.Ports
Asset.TOE.MonData	Attacker.Inet	Threat.SNMP.MITM	SecFunc.IPsec, SecFunc.SNMPv3
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.LAN	Threat.WEBconfig.Access	SecFunc.Auth.AdmCrds, SecFunc.Auth.AdmPwdChrs, SecFunc.Auth.BrtFrcCtr, SecFunc.Auth.AutoLogOut, SecFunc.Auth.Log
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.LAN	Threat.WEBconfig.MITM	SecFunc.HTTPS
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.LAN	Threat.CLI.Access	SecFunc.Auth.AdmCrds, SecFunc.Auth.AdmPwdChrs, SecFunc.Auth.BrtFrcCtr, SecFunc.Auth.AutoLogOut, SecFunc.Auth.Log
Asset.TOE.Config, Asset.TOE.MonData, Asset.TOE.Firmware	Attacker.LAN	Threat.CLI.MITM	SecFunc.SSH
Asset.TOE.MonData	Attacker.LAN	Threat.SNMP.Access	SecFunc.Auth.AdmCrds, SecFunc.Auth.AdmPwdChrs, SecFunc.Auth.BrtFrcCtr, SecFunc.Auth.AutoLogOut, SecFunc.Auth.Log
Asset.TOE.MonData	Attacker.LAN	Threat.SNMP.MITM	SecFunc.SNMPv3
Asset.User.Data.LAN	Attacker.Inet	Threat.LAN.Access	SecFunc.Firewall.Sessions, SecFunc.Firewall.DoS.IDS, SecFunc.Firewall.Log
Asset.User.Data.LAN	Attacker.Inet	Threat.IPsec.Access	SecFunc.IPsec, SecFunc.IPsec.Log, SecFunc.Firewall.Sessions, SecFunc.Firewall.DoS.IDS, SecFunc.Firewall.Log
Asset.User.Data.Inet	Attacker.Inet	Threat.IPsec.MITM	SecFunc.IPsec, SecFunc.IPsec.Log, SecFunc.Firewall.Sessions, SecFunc.Firewall.DoS.IDS, SecFunc.Firewall.Log

Limits of Evaluation

The following features of the TOE are out of scope for the evaluation:

- The TOE has a COM (serial) port that can be used to directly connect a computer to the TOE for management purposes.
- The TOE has an USB port that can be used for management purposes.
- The TOE provides IPv6 protocol services.

Appendix

Architecture Overview

The "LANCOM Business VPN Router 'LANCOM 1900EF' with LANCOM Systems Operating System 'LCOS 10.32.0029 PR' and IPsec VPN" (Target of Evaluation, TOE) consists of hardware (e.g. CPU, flash memory, memory) and software (OS).

The CPU is a 'Freescale QorIQ T1013 Communications Processor'. It contains the 'QorIQ Security Engine (SEC)' that provides cryptographic acceleration and offloading. The flash memory contains a file system that is created during production of the TOE.

The OS is the LANCOM Systems Operating System (LCOS). It is a proprietary (closed source) operating system not related to other well-known OSs (e.g. Microsoft Windows, Linux, macOS, BSD, UNIX). For the most part, it is developed by LANCOM Systems itself. This especially includes the HTTP, TLS, SSH, SNMPv3 and IPsec implementations.

LCOS implements part of the cryptographic algorithms and protocols using the OpenSSL crypto library. (Note: the OpenSSL SSL/TLS toolkit is not part of LCOS.) The OpenSSL crypto library is used to implement the DH, DHE, ECDH, ECDHE, ECDSA and RSA protocols, with the exception of the modular exponentiation.

LCOS implements part of the cryptographic algorithms and protocols using the SEC in the CPU. The SEC is used to implement the cipher algorithms (AES-GCM, AES-CTR, AES-CBC), the hash algorithms (SHA-256, SHA-384, SHA-512, SHA-1) and the corresponding HMAC algorithms, the modular exponentiation (e.g. for DH and RSA) and the random number generator (RNG).

The LCOS firmware is built as a statically linked monolithic binary and converted into a signed upload file. Uploading this file to the TOE replaces the LCOS firmware in the file system of the TOE as a whole (provided the signature can be validated). There are no mechanisms available to replace only parts of the LCOS firmware, or to add dynamic libraries or applications to it within the TOE.

The file system of the TOE is used internally by the LCOS firmware. There are no mechanisms available to access the file system of the TOE generally.

List of Libraries:

Library	Version
OpenSSL crypto library	1.0.2r

Update Mechanism

The Secure User Guidance (SUG) document describes how to update the LCOS firmware within the TOE using the WEBconfig management interface (during the initial configuration of the TOE).

The security of the update mechanism depends on the generation and verification of the signatures in the upload file containing the LCOS firmware. There are two RSA key pairs involved: An 8192-bit RSA Long-Term Key (LTK) pair and a 2048-bit RSA Firmware Signing Key (FSK) pair.

During the creation of the upload file, several SHA-256 hashes are calculated, signed with the private FSK and stored in the upload file. The public FSK is signed with the private LTK and stored in the upload file as well. All signatures are generated using RSA-PSS.

The TOE already contains the public LTK. After receiving the upload file, the TOE verifies the public FSK using the public LTK, verifies the SHA-256 hashes with the public FSK and calculates the SHA-256 hashes itself. If the SHA-256 hashes match, the upload file is accepted.

Cryptographic Specification

Cryptographic Mechanisms (TLS 1.2)

The following table specifies the cryptographic mechanisms that are used to enforce security functionality of the TOE:

Table of Cryptographic Mechanisms (TLS 1.2)			
1. Purpose	**2. Cryptographic Mechanism**	**3. Standard of Implementation**	**4. Key Size in Bits**
Authenticity	ECDSA signature generation and verification using SHA-2 (secp256r1, secp384r1, secp521r1)	[RFC 5246] (TLS), [RFC 8422] (TLSECC), [RFC 4366] (TLSEXT), [ANSI X9.62] (ECDSA), [SECG SEC2] (ECC), [QorIQ SEC] (RNG), [RFC 3280] (PKIX), [FIPS 180-4] (SHA)	|p| = 256, 384, 521
	RSA signature generation and verification (RSASSA-PKCS1-v1_5) using SHA-2	[RFC 5246] (TLS), [RFC 3447] (PKCS#1 v2.1), [RFC 3280] (PKIX), [FIPS 180-2] (SHA)	|n| = 2048, 3072, 4096
	- (Administrator action: Pre-configuration of administrator password on the server)	-	-
Authentication	ECDSA signature generation and verification using SHA-2 (secp256r1, secp384r1, secp521r1) (for ECDHE_ECDSA)	[RFC 5246] (TLS), [RFC 8422] (TLSECC), [RFC 4366] (TLSEXT), [ANSI X9.62] (ECDSA), [SECG SEC2] (ECC), [QorIQ SEC] (RNG), [RFC 3280] (PKIX), [FIPS 180-4] (SHA)	|p| = 256, 384, 521
	RSA signature generation and verification (RSASSA-PKCS1-v1_5) using SHA-2 (for ECDHE_RSA)	[RFC 5246] (TLS), [RFC 8422] (TLSECC), [RFC 4366] (TLSEXT), [RFC 8017] (PKCS#1 v2.2), [RFC 3280] (PKIX), [FIPS 180-4] (SHA)	|n| = 2048, 3072, 4096
	RSA signature generation and verification (RSASSA-PKCS1-v1_5) using SHA-2 (for DHE_RSA)	[RFC 5246] (TLS), [RFC 4366] (TLSEXT), [RFC 3447] (PKCS#1 v2.1), [RFC 3280] (PKIX), [FIPS 180-2] (SHA)	|n| = 2048, 3072, 4096
	Challenge-response password authentication using SHA-256	[FIPS 180-2] (SHA), [QorIQ SEC] (RNG)	|K| = 256
Key Agreement	ECDHE (secp256r1, secp384r1, secp521r1)	[RFC 8422] (TLSECC), [RFC 4366] (TLSEXT), [IEEE P1363] (ECDH), [SECG SEC2] (ECC), [QorIQ SEC] (RNG)	|p| = 256, 384, 521
	DHE	[RFC 5246] (TLS), [RFC 2631] (DH), [ANSI X9.42] (DH), [QorIQ SEC] (RNG), [RFC 3526] (MODP)	|p| = 2048, 3072,

Table of Cryptographic Mechanisms (TLS 1.2)			
			4096
Confidentiality	AES in GCM mode	[RFC 5246] (TLS), [RFC 5288] (AES-GCM), [RFC 5289] (AES-GCM), [RFC 5116] (AES-GCM), [FIPS 197] (AES), [SP 800-38D] (GCM)	|K| = 128, 256
	AES in CBC mode	[RFC 5246] (TLS), [FIPS 197] (AES), [SP 800-38A] (CBC), [QorIQ SEC] (RNG)	|K| = 128, 256
Integrity	AES in GCM mode	[RFC 5246] (TLS), [RFC 5288] (AES-GCM), [RFC 5289] (AES-GCM), [RFC 5116] (AES-GCM), [FIPS 197] (AES), [SP 800-38D] (GCM)	|K| = 128, 256
	HMAC with SHA-2	[RFC 5246] (TLS), [RFC 5289] (AES-CBC), [FIPS 180-2] (SHA), [RFC 2104] (HMAC)	|K| = 256, 384
Trusted Channel	TLS v1.2	[RFC 5246] (TLS), [RFC 5746] (TLS-RENEGO), [RFC 4086] (RANDOM), [ANSI X9.82] (RNG), [QorIQ SEC] (RNG)	-
Cryptographic Primitive	AES	[FIPS 197] (AES)	|K| = 128, 256
	SHA-256, SHA-384, SHA-512	[FIPS 180-2] (SHA)	-
	RNG	[QorIQ SEC] (RNG)	-

References (TLS 1.2)		
ANSI X9.42	Agreement Of Symmetric Keys Using Discrete Logarithm Cryptography	
ANSI X9.62	The Elliptic Curve Digital Signature Algorithm (ECDSA)	
ANSI X9.82	Random Number Generation	
FIPS 180-2	Secure Hash Standard (SHS)	https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf
FIPS 180-4	Secure Hash Standard (SHS)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-2	Digital Signature Standard (DSS)	https://csrc.nist.gov/csrc/media/publications/fips/186/2/archive/2000-01-27/documents/fips186-2.pdf
FIPS 197	Advanced Encryption Standard (AES)	https://nvlpubs.nist.gov/nistpubs/

References (TLS 1.2)		
		FIPS/NIST.FIPS.197.pdf
IEEE P1363	Standard Specifications for Public Key Cryptography	http://ieeexplore.ieee.org/document/891000/
QorIQ SEC	QorIQ T1024 Security (SEC) Reference Manual	
RFC 2104	HMAC: Keyed-Hashing for Message Authentication	https://tools.ietf.org/html/rfc2104
RFC 2631	Diffie-Hellman Key Agreement Method	https://tools.ietf.org/html/rfc2631
RFC 3280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	https://tools.ietf.org/html/rfc3280
RFC 3447	Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1	https://tools.ietf.org/html/rfc3447
RFC 3526	More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)	https://tools.ietf.org/html/rfc3526
RFC 4086	Randomness Requirements for Security	https://tools.ietf.org/html/rfc4086
RFC 4366	Transport Layer Security (TLS) Extensions	https://tools.ietf.org/html/rfc4366
RFC 5116	An Interface and Algorithms for Authenticated Encryption	https://tools.ietf.org/html/rfc5116
RFC 5246	The Transport Layer Security (TLS) Protocol Version 1.2	https://tools.ietf.org/html/rfc5246
RFC 5288	AES Galois Counter Mode (GCM) Cipher Suites for TLS	https://tools.ietf.org/html/rfc5288
RFC 5289	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)	https://tools.ietf.org/html/rfc5289
RFC 5746	Transport Layer Security (TLS) Renegotiation Indication Extension	https://tools.ietf.org/html/rfc5746
RFC 8017	PKCS #1: RSA Cryptography Specifications Version 2.2	https://tools.ietf.org/html/rfc8017
RFC 8422	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier	https://tools.ietf.org/html/rfc8422
SECG SEC1	SEC 1: Elliptic Curve Cryptography	https://www.secg.org/sec1-v2.pdf
SECG SEC2	SEC 2: Recommended Elliptic Curve Domain Parameters (Version 2.0)	https://www.secg.org/sec2-v2.pdf
SP 800-38A	Recommendation for Block Cipher Modes of Operation	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

Cryptographic Mechanisms (SSH)

The following table specifies the cryptographic mechanisms that are used to enforce security functionality of the TOE:

Table of Cryptographic Mechanisms (SSH)			
1. Purpose	**2. Cryptographic Mechanism**	**3. Standard of Implementation**	**4. Key Size in Bits**
Authenticity	- (Administrator action: Server host key fingerprint comparison on the client)	[RFC 4253] (SSH-TRANS)	-
	- (Administrator action: Pre-configuration of administrator public key on the server)	[RFC 4252] (SSH-USERAUTH)	-
	- (Administrator action: Pre-configuration of administrator password on the server)	-	-
Authentication	ECDSA signature generation and verification using SHA-2 (ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521)	[RFC 5656] (SSH-ECC), [SEC1] (ECC), [SEC2] (ECC), [ANSI X9.62] (ECDSA), [QorIQ SEC] (RNG), [FIPS 180-2] (SHA)	|p| = 256, 384, 521
	RSA signature generation and verification (RSASSA-PKCS1-v1_5) using SHA-2 (rsa-sha2-256, rsa-sha2-512)	[RFC 8332] (SSH-AUTH-SHA2), [RFC 4253] (SSH-TRANS), [RFC 4252] (SSH-USERAUTH), [RFC 8017] (PKCS#1 v2.2), [FIPS 180-4] (SHA)	|n| = 2048, 3072, 4096
Key Agreement	ECDH with SHA-2 (ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521)	[RFC 5656] (SSH-ECC), [SEC1] (ECC), [SEC2] (ECC), [ANSI X9.63] (ECC), [QorIQ SEC] (RNG), [FIPS 180-2] (SHA)	|p| = 256, 384, 521
	DH with SHA-2 (diffie-hellman-group-exchange-sha256)	[RFC 4419] (SSH-DH-GEX), [HAC] (DH), [QorIQ SEC] (RNG), [FIPS 180-2] (SHA)	|p| = 2048, 3072, 4096
Confidentiality	AES in GCM mode	[RFC 5647], [RFC 5116] (AES-GCM), [FIPS 197] (AES), [SP 800-38D] (GCM)	|K| = 128, 256
	AES in CTR mode	[RFC 4344], [FIPS 197] (AES), [SP 800-38A] (CTR)	|K| = 128, 192, 256
	AES in CBC mode	[RFC 4253] (SSH-TRANS), [FIPS 197] (AES), [SP 800-38A] (CBC), [QorIQ SEC] (RNG)	|K| = 128, 192, 256
Integrity	AES in GCM mode	[RFC 5647], [RFC 5116] (AES-GCM), [FIPS 197] (AES), [SP	|K| = 128,

Table of Cryptographic Mechanisms (SSH)			
		800-38D] (GCM)	256
	HMAC with SHA-2	[RFC 6668], [FIPS 180-2] (SHA), [RFC 2104] (HMAC)	|K| = 256, 512
Trusted Channel	SSH	[RFC 4251] (SSH-ARCH), [RFC 4252] (SSH-USERAUTH), [RFC 4253] (SSH-TRANS), [RFC 4254] (SSH-CONNECT), [QorIQ SEC] (RNG)	-
Cryptographic Primitive	AES	[FIPS 197] (AES)	|K| = 128, 192, 256
	SHA-256, SHA-384, SHA-512	[FIPS 180-2] (SHA)	-
	RNG	[QorIQ SEC] (RNG)	-

References (SSH)		
ANSI X9.62	The Elliptic Curve Digital Signature Algorithm (ECDSA)	
ANSI X9.63	Key Agreement and Key Transport using Elliptic Curve Cryptography	
FIPS 180-2	Secure Hash Standard (SHS)	https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf
FIPS 180-4	Secure Hash Standard (SHS)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-2	Digital Signature Standard (DSS)	https://csrc.nist.gov/csrc/media/publications/fips/186/2/archive/2000-01-27/documents/fips186-2.pdf
FIPS 197	Advanced Encryption Standard (AES)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
HAC	Handbook of Applied Cryptography	http://cacr.uwaterloo.ca/hac/
QorIQ SEC	QorIQ T1024 Security (SEC) Reference Manual	
RFC 2104	HMAC: Keyed-Hashing for Message Authentication	https://tools.ietf.org/html/rfc2104
RFC 2631	Diffie-Hellman Key Agreement Method	https://tools.ietf.org/html/rfc2631
RFC 4251	The Secure Shell (SSH) Protocol Architecture	https://tools.ietf.org/html/rfc4251
RFC 4252	The Secure Shell (SSH) Authentication Protocol	https://tools.ietf.org/html/rfc4252
RFC 4253	The Secure Shell (SSH) Transport Layer Protocol	https://tools.ietf.org/html/rfc4253

References (SSH)		
RFC 4254	The Secure Shell (SSH) Connection Protocol	https://tools.ietf.org/html/rfc4254
RFC 4256	Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)	https://tools.ietf.org/html/rfc4256
RFC 4344	The Secure Shell (SSH) Transport Layer Encryption Modes	https://tools.ietf.org/html/rfc4344
RFC 4419	Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol	https://tools.ietf.org/html/rfc4419
RFC 5116	An Interface and Algorithms for Authenticated Encryption	https://tools.ietf.org/html/rfc5116
RFC 5647	AES Galois Counter Mode for the Secure Shell Transport Layer Protocol	https://tools.ietf.org/html/rfc5647
RFC 5656	Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer	https://tools.ietf.org/html/rfc5656
RFC 6668	SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol	https://tools.ietf.org/html/rfc6668
RFC 8017	PKCS #1: RSA Cryptography Specifications Version 2.2	https://tools.ietf.org/html/rfc8017
RFC 8332	Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol	https://tools.ietf.org/html/rfc8332
SEC1	SEC 1: Elliptic Curve Cryptography	https://www.secg.org/sec1-v2.pdf
SEC2	SEC 2: Recommended Elliptic Curve Domain Parameters (Version 2.0)	https://www.secg.org/sec2-v2.pdf
SP 800-38A	Recommendation for Block Cipher Modes of Operation	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

Cryptographic Mechanisms (SNMPv3)

The following table specifies the cryptographic mechanisms that are used to enforce security functionality of the TOE:

Table of Cryptographic Mechanisms (SNMPv3)			
1. Purpose	**2. Cryptographic Mechanism**	**3. Standard of Implementation**	**4. Key Size in Bits**
Authenticity	- (Administrator action: Pre-configuration of authentication and privacy passwords on the server)	-	-
Authentication	-	-	-
Key Agreement	-	-	-
Confidentiality	AES in CFB mode	[RFC 3826] (SNMP-AES), [FIPS 197] (AES), [SP 800-38A] (CFB)	|K| = 128, 192, 256, s = 128
Integrity	HMAC with SHA-2	[RFC 7860] (SNMP-SHA-2), [RFC 6234] (SHA-2), [FIPS 180-4] (SHA), [RFC 2104] (HMAC)	|K| = 256, 384, 512
Trusted Channel	SNMPv3	[RFC 3411] (SNMP-ARCH), [RFC 3412] (SNMP-MSG), [RFC 3414] (SNMP-USM), [[RFC 3415] (SNMP-VACM)	-
Cryptographic Primitive	AES	[FIPS 197] (AES)	|K| = 128, 192, 256
	SHA-256, SHA-384, SHA-512	[FIPS 180-4] (SHA)	-
	RNG	[QorIQ SEC] (RNG)	-

References (SNMPv3)		
FIPS 180-4	Secure Hash Standard (SHS)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 197	Advanced Encryption Standard (AES)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
QorIQ SEC	QorIQ T1024 Security (SEC) Reference Manual	
RFC 2104	HMAC: Keyed-Hashing for Message Authentication	https://tools.ietf.org/html/rfc2104

References (SNMPv3)		
RFC 3411	An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks	https://tools.ietf.org/html/rfc3411
RFC 3412	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)	https://tools.ietf.org/html/rfc3412
RFC 3414	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)	https://tools.ietf.org/html/rfc3414
RFC 3415	View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)	https://tools.ietf.org/html/rfc3415
RFC 3826	The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model	https://tools.ietf.org/html/rfc3826
RFC 6234	US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)	https://tools.ietf.org/html/rfc6234
RFC 7860	HMAC-SHA-2 Authentication Protocols in User-Based Security Model (USM) for SNMPv3	https://tools.ietf.org/html/rfc7860
SP 800-38A	Recommendation for Block Cipher Modes of Operation	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

Cryptographic Mechanisms (IPsec)

The following table specifies the cryptographic mechanisms that are used to enforce security functionality of the TOE:

Table of Cryptographic Mechanisms (IPsec)			
1. Purpose	**2. Cryptographic Mechanism**	**3. Standard of Implementation**	**4. Key Size in Bits**
Authenticity	RSA signature generation and verification (RSASSA-PSS) using SHA-2	[RFC 7427] (IKEv2-SIGAUTH), [RFC 5280] (PKIX), [RFC 3447] (PKCS#1 v2.1), [FIPS 180-4] (SHA)	|n| = 2048, 3072, 4096
	- (Administrator action: Pre-configuration of pre-shared keys)	[RFC 7296] (IKEv2)	-
Authentication	RSA signature generation and verification (RSASSA-PSS) using SHA-2	[RFC 7427] (IKEv2-SIGAUTH), [RFC 5280] (PKIX), [RFC 3447] (PKCS#1 v2.1), [FIPS 180-4] (SHA)	|n| = 2048, 3072, 4096
	MAC generation and verification using pre-shared keys and SHA-2	[RFC 7296] (IKEv2), [FIPS 180-4] (SHA)	|K| = 256, 384, 512
Key Agreement	ECDH (secp256r1, secp384r1, secp521r1)	[RFC 5903] (IKE-ECP), [IEEE P1363] (ECDH), [SECG SEC2] (ECC), [QorIQ SEC] (RNG)	|p| = 256, 384, 521
	DH	[RFC 2631] (DH), [ANSI X9.42] (DH), [QorIQ SEC] (RNG), [RFC 3526] (MODP)	|p| = 2048, 3072, 4096
Confidentiality	AES in GCM mode	[RFC 5282], [RFC 5116], [RFC 4106], [FIPS 197] (AES), [SP 800-38D] (GCM)	|K| = 128, 192, 256
	AES in CBC mode	[RFC 3602], [FIPS 197] (AES), [SP 800-38A] (CBC), [QorIQ SEC] (RNG)	|K| = 128, 192, 256
Integrity	AES in GCM mode	[RFC 5282], [RFC 5116], [RFC 4106], [FIPS 197] (AES), [SP 800-38D] (GCM)	|K| = 128, 192, 256
	HMAC with SHA-2	[RFC 4868], [FIPS 180-2] (SHA), [RFC 2104] (HMAC)	|K| = 256, 384,

Table of Cryptographic Mechanisms (IPsec)			
			512
Trusted Channel	IPsec with IKEv2 and ESP	[RFC 4301] (IPsec), [RFC 7296] (IKEv2), [RFC 8247] (ALGO-IKE), [RFC 4303] (ESP), [RFC 8221] (ALGO-ESP), [RFC 4086] (RANDOM), [ANSI X9.82] (RNG), [QorIQ SEC] (RNG)	-
Cryptographic Primitive	AES	[FIPS 197] (AES)	|K| = 128, 192, 256
	SHA-256, SHA-384, SHA-512, SHA-1	[FIPS 180-2] (SHA)	-
	RNG	[QorIQ SEC] (RNG)	-

References (IPsec)		
ANSI X9.42	Agreement Of Symmetric Keys Using Discrete Logarithm Cryptography	
ANSI X9.82	Random Number Generation	
FIPS 180-2	Secure Hash Standard (SHS)	https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf
FIPS 180-4	Secure Hash Standard (SHS)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 197	Advanced Encryption Standard (AES)	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
IEEE P1363	Standard Specifications for Public Key Cryptography	http://ieeexplore.ieee.org/document/891000/
QorIQ SEC	QorIQ T1024 Security (SEC) Reference Manual	
RFC 2104	HMAC: Keyed-Hashing for Message Authentication	https://tools.ietf.org/html/rfc2104
RFC 2631	Diffie-Hellman Key Agreement Method	https://tools.ietf.org/html/rfc2631
RFC 3447	Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1	https://tools.ietf.org/html/rfc3447
RFC 3526	More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)	https://tools.ietf.org/html/rfc3526
RFC 3602	The AES-CBC Cipher Algorithm and Its Use with IPsec	https://tools.ietf.org/html/rfc3602
RFC 4086	Randomness Requirements for Security	https://tools.ietf.org/html/rfc4086
RFC 4106	The Use of Galois/Counter Mode (GCM) in IPsec	https://tools.ietf.org/html/rfc4106

References (IPsec)		
	Encapsulating Security Payload (ESP)	
RFC 4301	Security Architecture for the Internet Protocol	https://tools.ietf.org/html/rfc4301
RFC 4303	IP Encapsulating Security Payload (ESP)	https://tools.ietf.org/html/rfc4303
RFC 4868	Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec	https://tools.ietf.org/html/rfc4868
RFC 5116	An Interface and Algorithms for Authenticated Encryption	https://tools.ietf.org/html/rfc5116
RFC 5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	https://tools.ietf.org/html/rfc5280
RFC 5282	Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol	https://tools.ietf.org/html/rfc5282
RFC 5903	Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2	https://tools.ietf.org/html/rfc5903
RFC 7296	Internet Key Exchange Protocol Version 2 (IKEv2)	https://tools.ietf.org/html/rfc7296
RFC 7427	Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)	https://tools.ietf.org/html/rfc7427
RFC 8221	Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)	https://tools.ietf.org/html/rfc8221
RFC 8247	Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)	https://tools.ietf.org/html/rfc8247
SEC2	SEC 2: Recommended Elliptic Curve Domain Parameters (Version 2.0)	https://www.secg.org/sec2-v2.pdf
SP 800-38A	Recommendation for Block Cipher Modes of Operation	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

Random Source Description

Random numbers for cryptographic algorithms and protocols are created by the RNG of the SEC in the CPU. Details are provided in the 'QorIQ T1024 Security (SEC) Reference Manual' in chapter '2 Feature summary' and section '11.6 Random-number generator (RNG) functionality'. The RNG utilizes a true random-number generator (TRNG) as well as a pseudo random-number generator (PRNG) to achieve both true randomness and cryptographic strength. The PRNG uses DRBG-HASH (NIST 800-90) with SHA-256, and it is seeded by the TRNG (free running ring oscillator) with 512 bits of high-grade random entropy.

Key Management Description

Cryptographic Keys and Parameters (TLS 1.2 Server, HTTP Server)

The following cryptographic keys and parameters are used by the TOE for cryptographic operations:

- **Server RSA public/private key pair** - A 2048-bit RSA key pair, consisting of an RSA private key and an RSA public key. It is generated automatically when the TOE configuration is reset (e.g. during commissioning and decommissioning), according to [RFC 3447] (PKCS#1 v2.1), section 3, and [QorIQ SEC] (RNG), section 11.6. It is used to calculate the server root certificate and the server on-the-fly certificates. The RSA private key is used to authenticate the server to the client, by generating a signature per connection and sending it to the client, according to [RFC 8422] (TLSECC), section 5.4, or [RFC 5246] (TLS), section 7.4.3. The RSA key pair is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Server root certificate** - A self-signed X.509 public key certificate, binding the server RSA public key to the serial number of the TOE. It is generated automatically using the server RSA public/private key pair, when the TOE is powered on or restarted, according to [RFC 3280] (PKIX). It is used together with the server on-the-fly certificates. It is held in memory, and destroyed when the TOE is powered off.
- **Server on-the-fly certificate** - An X.509 public key certificate, binding the server RSA public key to the server domain name or the server IP address, with that the client is negotiating the connection. It is generated automatically using the server RSA public/private key pair and the server root certificate, when a connection is negotiated, according to [RFC 3280] (PKIX). It is used together with the server root certificate and the server RSA private key to authenticate the server to the client, according to [RFC 8422] (TLSECC), section 5.3, or [RFC 5246] (TLS), section 7.4.2. It is held in memory, and destroyed when the TOE is powered off.
- **Client random** - A client-generated random structure, containing a 4-byte timestamp and 28 random bytes, received from the client. It is used and destroyed together with the server random.
- **Server random** - A server-generated random structure, containing a 4-byte timestamp and 28 random bytes, generated per connection by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. Client random and server random are used together in the server authentication to bind the signature to the current negotiation. They are used together in the calculation of the master secret and the keying material (cf. MAC keys, encryption keys and/or IVs). They are held within the connection state in memory, and overwritten with zeros, when the connection is closed.
- **Server ECDH/DH private key** - A Diffie-Hellman private key, depending on the negotiated cipher suite. It is generated per connection by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. It is used to calculate the server ECDH/DH public key and the premaster secret. It is held in memory, and overwritten with zeros, when the premaster secret has been calculated.
- **Server ECDH/DH public key** - A Diffie-Hellman public key, depending on the negotiated cipher suite. It is calculated from the server ECDH/DH private key, according to [RFC 8422] (TLSECC), section 5.4, or [RFC 5246] (TLS), section 7.4.3. It is sent to the client to enable the client to calculate the premaster secret. It is held in memory, and overwritten with zeros, when the premaster secret has been calculated.

- **Client ECDH/DH public key** - A Diffie-Hellman public key, depending on the negotiated cipher suite. It is received from the client. It is used to calculate the premaster secret. It is held in memory, and overwritten with zeros, when the premaster secret has been calculated.
- **Premaster secret** - A secret shared between the client and the server in a connection. It is calculated from the client ECDH/DH public key and the server ECDH/DH private key, according to [RFC 5246] (TLS), section 8.1.2. It is used to calculate the master secret. It is held in memory, and overwritten with zeros, when the master secret has been calculated.
- **Master secret** - A 48-byte secret shared between the client and the server in a connection. It is calculated from the premaster secret, the client random and the server random, according to [RFC 5246] (TLS), section 8.1. It is used to calculate the keying material (cf. MAC keys, encryption keys and/or IVs). It is held within the connection state in memory, and overwritten with zeros, when the connection is closed. Additionally, it is held within the TLS session state in memory for 10 to 20 minutes to facilitate TLS session resumption, and overwritten with zeros afterwards.
- **MAC keys, encryption keys, IVs** - Keys and parameters shared between the client and the server in a connection. Depending on the negotiated cipher suite, the necessary MAC keys, encryption keys and/or IVs are calculated from the master secret, the client random and the server random, according to [RFC 5246] (TLS), section 6.3. They are used by the TLS record protocol to provide privacy and data integrity for the application data protocol (e.g. HTTP). They are held within the connection state in memory, and overwritten with zeros, when the connection is closed.
- **Administrator password** - A password shared between the administrator and the TOE. The administrator generates their password during commissioning of the TOE (according to the SUG). It is used inside the TLS connection to authenticate the administrator to the HTTP server. It is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **HTTP server challenge** - A server-generated 256-bit random value, generated per HTTP session login by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. It is sent to the HTTP client as part of the HTTP server login page, to enable the HTTP client to calculate the HTTP client response to authenticate the administrator to the HTTP server. It is held in memory, and destroyed either after receiving and evaluating a HTTP client response or after 120 seconds. Additionally, after a successful login it is held within the HTTP session state in memory, and destroyed when the HTTP session is terminated (cf. HTTP session ID).
- **HTTP client response** - A 256-bit keyed hash, generated per HTTP session login. The HTTP client calculates the SHA-256 hash of the concatenation of the received HTTP server challenge and the password entered into the HTTP server login page on the HTTP client, and sends it to the HTTP server. The HTTP server calculates the SHA-256 hash of the concatenation of the HTTP server challenge and the administrator password, and compares it to the received SHA-256 hash. If the SHA-256 hashes match, the login is accepted. The SHA-256 hashes are held in memory, and overwritten with zeros after the comparison. Additionally, after a successful login the SHA-256 hash is held within the HTTP session state in memory, and destroyed when the HTTP session is terminated (cf. HTTP session ID).
- **HTTP session ID** - A server-generated 256-bit hash value, identifying an authenticated HTTP session. It is generated per HTTP session by calculating the SHA-256 hash of the sum of a 160-bit random value and the concatenation of the HTTP clients IPv4 address, the current time and the next-hop MAC address. The random value is generated by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. The HTTP session ID is sent to the

HTTP client within URLs as a URL parameter, to enable the HTTP client to request HTTP server pages without additional authentication. It is held within the HTTP session state in memory, and destroyed together with the HTTP session state when the HTTP session is terminated, either when the HTTP server logout is requested by the administrator pressing the corresponding button on a HTTP server page, or when the configurable HTTP session inactivity timeout expires (default: 10 minutes).

Cryptographic Keys and Parameters (SSH Server)

The following cryptographic keys and parameters are used by the TOE for cryptographic operations:

- **Server ECDSA public/private host key pair** - A 256-bit ECDSA key pair, consisting of an ECDSA private key and an ECDSA public key. It is generated automatically when the TOE configuration is reset (e.g. during commissioning and decommissioning), according to [SEC1] (ECC), section 3.2, and [QorIQ SEC] (RNG), section 11.6. Depending on the negotiated public key algorithm, the ECDSA key pair is used as the server host key pair. The ECDSA private key is used to authenticate the server to the client, by generating a signature per connection and sending it to the client, together with the ECDSA public key, according to [RFC 5656] (SSH-ECC), sections 3 and 4. The ECDSA key pair is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Server RSA public/private host key pair** - A 2048-bit RSA key pair, consisting of an RSA private key and an RSA public key. It is generated automatically when the TOE configuration is reset (e.g. during commissioning and decommissioning), according to [RFC 8017] (PKCS#1 v2.2), section 3, and [QorIQ SEC] (RNG), section 11.6. Depending on the negotiated public key algorithm, the RSA key pair is used as the server host key pair. The RSA private key is used to authenticate the server to the client, by generating a signature per connection and sending it to the client, together with the RSA public key, according to [RFC 4419] (SSH-DH-GEX), section 3, and [RFC 8332] (SSH-AUTH-SHA2), section 3. The RSA key pair is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Client cookie** - A client-generated 16-byte random value, received from the client. It is used and destroyed together with the server cookie.
- **Server cookie** - A server-generated 16-byte random value, generated per connection by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. Client cookie and server cookie are used together in the calculation of the exchange hash, to bind the server authentication signature and keying material derived from the exchange hash to the current negotiation. They are held within the connection state in memory, and overwritten with zeros, when the connection is closed.
- **Server ECDH/DH private key** - A Diffie-Hellman private key, depending on the negotiated key exchange method. It is generated per connection by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. It is used to calculate the server ECDH/DH public key and the ECDH/DH shared secret. It is held in memory, and overwritten with zeros, when the ECDH/DH shared secret has been calculated.
- **Server ECDH/DH public key** - A Diffie-Hellman public key, depending on the negotiated key exchange method. It is calculated from the server ECDH/DH private key, according to [RFC 5656] (SSH-ECC), section 4, or [[RFC 4419] (SSH-DH-GEX), section 3. It is sent to the client to enable the client to calculate the ECDH/DH shared secret. It is held in memory, and overwritten with zeros, when the ECDH/DH shared secret has been calculated.
- **Client ECDH/DH public key** - A Diffie-Hellman public key, depending on the negotiated key exchange method. It is received from the client. It is used to calculate the ECDH/DH shared secret. It is held in memory, and overwritten with zeros, when the ECDH/DH shared secret has been calculated.
- **ECDH/DH shared secret** - A Diffie-Hellman shared secret, depending on the negotiated key exchange method. It is calculated from the client ECDH/DH public key and the server ECDH/DH

private key, according to [RFC 5656] (SSH-ECC), section 4, or [[RFC 4419] (SSH-DH-GEX), section 3. It is used in the calculation of the exchange hash, and to calculate the keying material (cf. Initial IVs, encryption keys and/or MAC keys). It is held in memory, and overwritten with zeros, when the exchange hash and the keying material have been calculated.

- **Exchange hash (session identifier)** - A secret shared between the client and the server in a connection, and used as the session identifier. It is calculated from the ECDH/DH shared secret, the client and server ECDH/DH public keys, the server public host key, and other data used in the current negotiation (including the client and server cookies), according to [RFC 5656] (SSH-ECC), section 4, or [[RFC 4419] (SSH-DH-GEX), section 3. It is used to calculate the keying material (cf. Initial IVs, encryption keys and/or MAC keys). It is held in memory, and overwritten with zeros, when the keying material has been calculated. Additionally, it is held within the connection state in memory as the session identifier, and overwritten with zeros, when the connection is closed.
- **Initial IVs, encryption keys and/or MAC keys** - Keys and parameters shared between the client and the server in a connection. Depending on the negotiated encryption and message authentication algorithms, the necessary initial IVs, encryption keys and/or MAC keys are calculated from the ECDH/DH shared secret and the exchange hash, according to [RFC 4253] (SSH-TRANS), section 7.2. They are used by the SSH transport layer to provide strong encryption and integrity protection to the transported messages. They are held within the connection state in memory, and overwritten with zeros, when the connection is closed.
- **Administrator ECDSA/RSA public key** - An ECDSA/RSA public key shared between the administrator and the TOE. The administrator generates an ECDSA/RSA public/private key pair during commissioning of the TOE, and adds the ECDSA/RSA public key to the accepted public keys within the TOE configuration (according to the SUG). It is used to authenticate the administrator to the server, by verifying a signature received from the client, when using the public-key client authentication method, according to [RFC 4252] (SSH-USERAUTH), section 7. It is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Administrator password** - A password shared between the administrator and the TOE. The administrator generates their password during commissioning of the TOE (according to the SUG). It is used to authenticate the administrator to the server, when using either the password or the keyboard-interactive client authentication method, according to [RFC 4252] (SSH-USERAUTH), section 8, or [RFC 4256] (SSH-KBDINT), section 3. It is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).

Cryptographic Keys and Parameters (SNMPv3 Server)

The following cryptographic keys and parameters are used by the TOE for cryptographic operations:

- **Authentication password** - A password shared between the administrator and the TOE. The administrator generates their password during commissioning of the TOE (according to the SUG). It is used to calculate the authentication key. It is held within the TOE configuration in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Privacy password** - A password shared between the administrator and the TOE. The administrator generates their password during commissioning of the TOE (according to the SUG). It is used to calculate the privacy key. It is held within the TOE configuration in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Authentication key** - A key shared between the client and the server. It is calculated from the authentication password only when needed, according to [RFC 7860] (SNMP-SHA-2), section 5, and [RFC 3414] (SNMP-USM), section 2.6. It is used by the security subsystem of the user-based security model to provide authentication to transported messages. It is held in memory, and destroyed when the TOE is powered off.
- **Privacy key** - A key shared between the client and the server. It is calculated from the privacy password only when needed, according to [RFC 3826] (SNMP-AES), section 1.2, and [RFC 3414] (SNMP-USM), section 2.6. It is used by the security subsystem of the user-based security model to provide privacy to transported messages. It is held in memory, and destroyed when the TOE is powered off.
- **Initialization vector salt** - A server-generated 8-byte random value, generated only when needed by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. It is used as part of the initialization vector, according to [RFC 3826] (SNMP-AES), section 3.1.2. It is held in memory, and destroyed when the TOE is powered off.

Cryptographic Keys and Parameters (IPsec Initiator/Responder)

The following cryptographic keys and parameters are used by the TOE for cryptographic operations:

- **Local RSA public/private key pair and certificate** - A 2048/3072/4096-bit RSA key pair, consisting of an RSA private key and an RSA public key within an RSA certificate. The administrator adds a PKCS#12 file to the TOE configuration, containing an RSA private key, an RSA certificate binding the corresponding RSA public key to one or more identities of the TOE (e.g. IPv4 addresses, domain names, email addresses), and trusted RSA CA certificates needed to verify the RSA certificate, if applicable. The RSA private key is used to authenticate the TOE to the peer, by generating a signature per connection and sending it to the client, together with the RSA certificate, when using either the digital signature authentication method or the RSA signature authentication method, according to [RFC 7427] (IKEv2-SIGAUTH), section 3, and/or [RFC 7296] (IKEv2), section 2.15. The PKCS#12 file is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Trusted RSA certificates** - RSA CA or self-signed certificates trusted by the administrator and the TOE. The administrator adds a PKCS#12 file to the TOE configuration, containing trusted RSA CA and/or self-signed certificates (e.g. a self-signed remote RSA certificate). The RSA certificates are used to authenticate the peer, by verifying a remote RSA certificate and a signature both received from the peer, when using either the digital signature authentication method or the RSA signature authentication method, according to [RFC 7427] (IKEv2-SIGAUTH), section 3, and/or [RFC 7296] (IKEv2), section 2.15. The PKCS#12 file is held in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Local/remote pre-shared keys** - Two passwords (typically chosen the same) shared between the peer and the TOE in a connection. The administrator generates the pre-shared keys per peer, and adds them to the IPsec VPN peer configuration within the TOE configuration. The local pre-shared key is used to authenticate the TOE to the peer, by generating a signature per connection and sending it to the client, and the remote pre-shared key is used to authenticate the peer to the TOE, by verifying a signature per connection received from the peer, when using the pre-shared key authentication method, according to [RFC 7296] (IKEv2), section 2.15. They are held within the TOE configuration in flash memory, and permanently removed from the flash memory when decommissioning the TOE (according to the SUG).
- **Peer nonce** - A peer-generated random value, received from the peer. It is used and destroyed together with the initiator/responder nonce.
- **Initiator/responder nonce** - A 256-bit random value, generated per connection by a secure random number generator, according to [RFC 7296] (IKEv2), section 2.10, and [QorIQ SEC] (RNG), section 11.6. The peer nonce and the initiator/responder nonce are used together in the calculation of the keying material for the IKE SA and the Child SAs, and in the authentication of the IKE SA. They are held within the connection states (IKE/Child SAs) in memory, and destroyed when the connections are either rekeyed or closed.
- **Initiator/responder ECDH/DH private** key - A Diffie-Hellman private key, depending on the negotiated Diffie-Hellman group. It is generated per connection by a secure random number generator, according to [QorIQ SEC] (RNG), section 11.6. It is used to calculate the initiator/responder ECDH/DH public key and the ECDH/DH shared secret. It is held in memory, and overwritten with zeros, when the ECDH/DH shared secret has been calculated.
- **Initiator/responder ECDH/DH public** key - A Diffie-Hellman public key, depending on the negotiated Diffie-Hellman group. It is calculated from the initiator/responder ECDH/DH private

key, according to [RFC 7296] (IKEv2), sections 1.2 and 1.3. It is sent to the peer to enable the peer to calculate the ECDH/DH shared secret. It is held in memory, and overwritten with zeros, when the ECDH/DH shared secret has been calculated.

- **Peer ECDH/DH public key** - A Diffie-Hellman public key, depending on the negotiated Diffie-Hellman group. It is received from the peer. It is used to calculate the ECDH/DH shared secret. It is held in memory, and overwritten with zeros, when the ECDH/DH shared secret has been calculated.
- **ECDH/DH shared secret** - A Diffie-Hellman shared secret, depending on the negotiated Diffie-Hellman group. It is calculated from the peer ECDH/DH public key and the initiator/responder ECDH/DH private key, according to [RFC 7296] (IKEv2), sections 1.2 and 1.3. It is used in the calculation of the keying material for the IKE SA and the Child SAs. It is held in memory, and overwritten with zeros, when the keying material has been calculated.
- **IKE SA keying material** - Keying material and keys shared between the peer and the initiator/responder in a connection. Depending on the negotiated encryption algorithm, integrity protection algorithm and pseudorandom function, the necessary keying material and keys are calculated from the peer and initiator/responder nonces and the ECDH/DH shared secret, according to [RFC 7296] (IKEv2), section 2.14. They are used to provide confidentiality, integrity protection and authenticity to subsequent negotiations (e.g. Child SA negotiations), and to calculate the Child SA keying material. They are held within the connection state (IKE SA) in memory, and overwritten with zeros, when the connection is either rekeyed or closed.
- **Child SA keying material** - Keying material and keys shared between the peer and the initiator/responder in a connection. Depending on the negotiated encryption algorithm, integrity protection algorithm and pseudorandom function, the necessary keying material and keys are calculated from the IKE SA keying material, the peer and initiator/responder nonces and the ECDH/DH shared secret, according to [RFC 7296] (IKEv2), section 2.17. They are used to provide confidentiality, integrity protection and authenticity to transported data. They are held within the connection state (Child SA) in memory, and overwritten with zeros, when the connection is either rekeyed or closed.