Chelpis PQTunnel v1.1.5

Security Target

Evaluation Assurance Level (EAL): EAL1

Doc Ref: PQ-001
Doc Ver: 1.9
Date: 29th January 2024

Developed by:

Prepared by:

Version History

Version	Date	Description
0.1	21st June 2023	Initial draft.
0.2	1st July 2023	Updated based on developer feedback.
0.3	11th July 2023	Updated based on developer feedback.
0.4	24th July 2023	Updated based on Laboratory feedback.
1.0	28th July 2023	Submitted to Laboratory for evaluation.
1.1	14th August 2023	Updated based on Observation Report (OR-001-d1).
1.2	23rd August 2023	Updated based on Observation Report (OR-001-d2).
1.3	29th September 2023	Updated based on Observation Report (OR-005-d1). • Modified "PQ-Tunnel" to "PQTunnel" • Added Section **1.3**, **1.4**, **1.7.3 (1.7.3.4**, **1.7.3.5**, **1.7.3.6)**, **5.2.4 (5.2.4.1**, **5.2.4.2)**, **5.2.5 (5.2.5.1)**, 5.**2.6 (5.2.6.1)**, **6 (6.4, 6.5, 6.6)**. • Modified **Figure 1** and **TOE Description in 1.7**. • Added SFR **FMT_SMR.1**, **FMT_SMF.1**, **FAU_SAR.1**, and **FTA_SSL.4**. • Added **OE.TIMESTAMP** in Section 3.1.
1.4	16th October 2023	Updated based on Observation Report (OR-005-d2), (OR-006-d1) and (OR-007-d1). • Updated TOE name to "PQTunnel" and version to v1.1.5 • Changed evaluation platform to MacOS • Modified Section 5.2.2.4 and 6.2 to incorporate "AES keys" in AEAD cryptography.
1.5	16th November 2023	Updated based on Observation Report (OR-008-d1). • Updated section 1.7.2 to include the version of PQTunnel Installation Manual (v1.0.1). • Modified FCS_CKM.1(4) naming to maintain consistency. • "Account ID" has been modified to "Account" to maintain consistency with the TOE. • Modified "Multi-Factor Authentication" to "Two-Factor Authentication"; and "MFA" to "2FA" to maintain consistency with the TOE. • Modified Section 5.2.4.2 to distinguish between the security function for "Password" and "2FA", due to similarity in the parameters. • Updated Section 5.2.5.1 which has added "operations" into the assignments.
1.6	4th December 2023	Updated based on Observation Report (OR-010-d1). • Added explanation on the [Device Interface] non-TOE component. • Revised the [Abbreviation and Definition] list.
1.7	12th December 2023	Updated based on Observation Report (OR-010-d2) and (OR-011-d1).
1.8	2nd January 2024	Updated the standards for some of the cryptographic key generation and operations.

		• Section 5.2.2.1:
		  ○ From "*FIPS PUB 186-5, Appendix A.2*" to "*FIPS PUB 186-4, Appendix B.4*".
		  ○ "*FIPS SP 800-186, Appendix G*" has been removed as it's redundant.
		• Section 5.2.2.2:
		  ○ "CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.01) January 31, 2021 (https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf)" has been removed.
		  ○ "The Open Quantum Safe Project (https://openquantumsafe.org/)" has been removed.
		  ○ "Post-Quantum Cryptography (CRYSTALS-KYBER) (https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions)" has been added.
		• Section 5.2.2.3
		  ○ "*FIPS SP 800-186, Appendix G*" has been removed as it's redundant.
		• Section 5.2.2.7:
		  ○ From "*FIPS PUB 186-5, Section 6.4, ECDSA Digital Signature Generation and Verification*" to "*FIPS PUB 186-4, Section 6.4, ECDSA Digital Signature Generation and Verification*".
		  ○ "*FIPS SP 800-186, Appendix G*" has been removed as it's redundant.
		• Section 5.2.2.8:
		  ○ "CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.01) January 31, 2021 (https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf)" has been removed.
		  ○ "The Open Quantum Safe Project (https://openquantumsafe.org/)" has been removed.
		  ○ "Post-Quantum Cryptography (CRYSTALS-KYBER) (https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions)" has been added.
		• Section 5.2.2.9:
		  ○ "*FIPS SP 800-186, Appendix G*" has been removed as it's redundant".
		• Section 5.2.2.10:
		  ○ From "*RFC 7539*" to "*RFC 8439*".

1.9	29th January 2024	Updated based on Observation Report (OR-013-d1).

Table of Contents

List of Figures

List of Tables

1 SECURITY TARGET INTRODUCTION

This security target (ST) defines the scope of the evaluation in terms of the assumptions made, the intended environment for the Target of Evaluation (TOE), the Information Technology (IT) security functional and assurance requirements to be met, and the level of confidence (evaluation assurance level) to which it is asserted that the TOE satisfies its IT security requirements. This document forms the baseline for the Common Criteria (CC) evaluation.

1.1 SECURITY TARGET REFERENCE

ST Title: Chelpis PQTunnel v1.1.5
ST Version: 1.9
ST Date: 29th January 2024

1.2 TOE REFERENCE

TOE Name: PQTunnel
TOE Version: 1.1.5
Developer Name: Chelpis Quantum Tech Co. Ltd.

1.3 GLOSSARY

Table 1: Glossary of terms used in this evaluation.

Terms	Description
UserAgent	This represents the TOE. The UserAgent authenticates with the Console and attempts to establish a tunnel with a specific ServiceHostAgent.
UserAgent/device	This term is used interchangeably with "Platform", which represents any system (e.g., computer) the user logs into. After a user authenticates with the Console through the TOE on any computer, the computer is registered to the user's device list.
ServiceHostAgent	Also known as Endpoint or Host, which are endpoints that the user wants to communicate with remotely The agent is installed on ServiceHostAgent/device to be activated and allow tunnel establishment to occur.
ServiceHostAgent/device	This represents the underlying platform (e.g., computer, server, databases, etc.) of the Endpoint or Host. After an administrator authenticates with the Console using a ServiceHostAgent, the computer is registered to the device list of that service server.

1.4 ABBREVIATIONS AND DEFINITIONS

Table 2: Common acronyms and their definitions.

Abbreviation	Definition
ST	Security Target
TOE	Target of Evaluation
IT	Information Technology
CC	Common Criteria
SFR	Security Functional Requirement
SAR	Security Assurance Requirement
TSF	TOE Security Functionality
TSFI	TSF Interface

EAL	Evaluation Assurance Level
IP	Internet Protocol
OS	Operating System
PP	Protection Profile
NIST	National Institute of Standards and Technology
E2EE	End-to-End Encryption
CRUD	Create, Read, Update and Delete
PQ	Post-Quantum
PQC	Post-Quantum Cryptography
VPN	Virtual Private Network
KEM	Key Encapsulation Mechanism
2FA	Two-Factor Authentication
JWT	JSON Web Token
UUID	Universally Unique Identifier
TLS	Transport Layer Security
TOTP	Time-Based One Time Password
RBG	Random Bit Generator
RNG	Random Number Generator
ECC	Elliptical Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
HMAC	Hash-based Message Authentication Code
HKDF	HMAC-based Key Derivation Function
G2FA	Google Two-Factor Authenticator (Google Authenticator)

1.5 TOE TYPE

PQTunnel v1.1.5 is a next-generation Virtual Private Network (VPN), a software application which provides end-to-end secure tunnelling channels that are resistant to quantum attacks from the user's device to respective backend resources. The TOE provides protection of data in transit across a shared or public network. The TOE employs quantum-resistant cryptographic algorithms that are recognised by National Institute of Standards and Technology (NIST).

1.6 TOE OVERVIEW

The TOE is PQTunnel v1.1.5, which is a software application and a next-generation, Client-to-Client Virtual Private Network (VPN). The TOE provide connections based on Post-Quantum Cryptography (PQC) with Zero Trust Architecture to provide an E2EE secure tunnelling channels from the user's device to respective backend resources that are resistant to quantum attacks. The TOE is owned and developed by the Chelpis Quantum Tech Co. Ltd. since September 2021.

The TOE may also be referred to as "UserAgent".

From this point onwards, any terms such as "User(s)", "user(s)", "authorised user(s)", or "local user(s)" are referring to the TOE user. In addition, any terms such as "Administrator(s)", "administrator(s)", "Admin(s)" or "admin(s)" are referring to the Console Administrator. The Console Administrator is not included in the scope of this Evaluation.

Compared to traditional VPNs, the TOE utilizes NIST PQC-KEM (Post-Quantum Cryptographic Key Encapsulation Mechanism) to assist in establishing a shared secret between two endpoints. This approach effectively protects against quantum computer attacks. Moreover, the TOE prevent security incidents such as lateral movement through zero trust architecture and provide virtual IPs for each backend resources. The TOE can also limit the exposure in case of security exploitations due to the micro-segmentation of the backend resources and multi-layered identity authentication.

The key features of the TOE are listed as below:

- Support quantum-resistant secure channels for users to communicate remotely with their targeted endpoints;
- Support various deployment environment such as on-premise or on cloud;
- Provide direct, end-to-end connections between user devices and backend resources;
- Micro-segmentation of backend resources which reduces the impacted range by cyber-attacks such as malwares and ransomwares; and
- Two-Factor Authentication (2FA) which provide additional layer of security to access.

1.7 TOE DESCRIPTION

The TOE consists of the PQTunnel v1.1.5. Figure 1 shows the evaluated configuration of these TOE components, which reflects a typical implementation configuration.

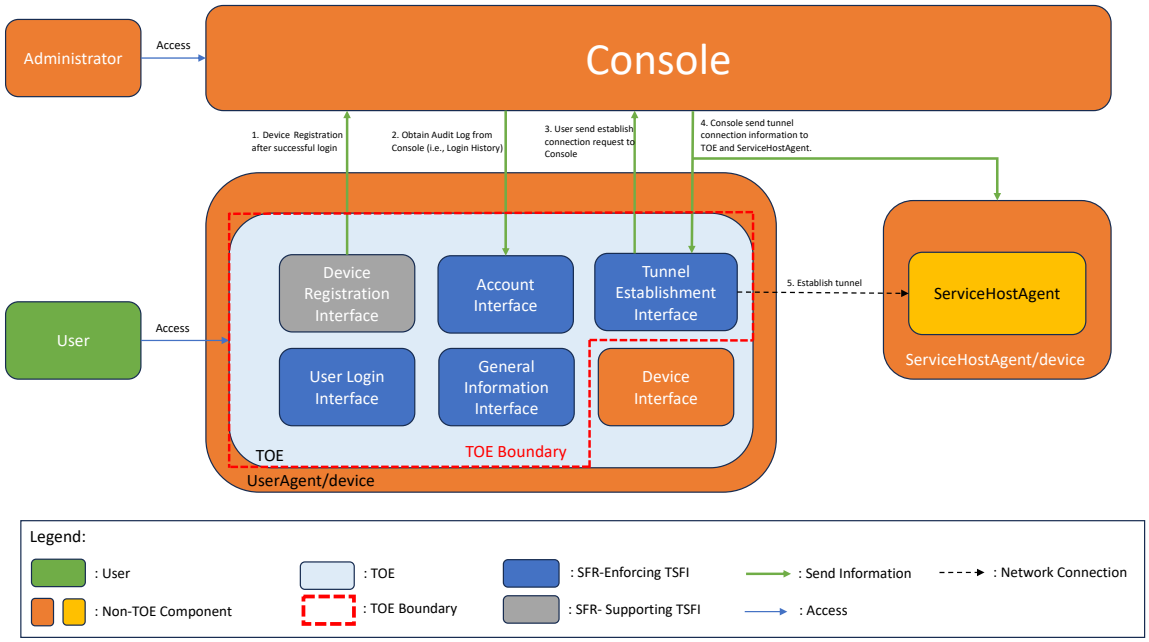

Figure 1: TOE boundary

Figure 1 shows the implementation diagram of how the TOE is accessed and utilised to perform tunnel establishment. The TOE is installed on the user device (i.e., platform or UserAgent/device) such as desktop, laptop and so on. Next, the Console is accessible through a browser and is located on the cloud, it's used to manage the access control policy of the TOE. However, only the administrator have access to the Console and this evaluation do not include the Console into the scope. Lastly, the ServiceHostAgent (i.e., endpoint or host) is installed on

backend resources (i.e., ServiceHostAgent/device) such as computer, server, databases, etc. to verify and accept connection establishment from the TOE.

Before the users can perform actions on the TOE, they are required to authenticate themselves. The users are required to provide their Account, Password and Domain. After successfully authenticating themselves, the TOE will accept a JWT session token from the Console. In addition, the TOE will generate a unique identifier (i.e., UUID v4) for the device used by the user, and an asymmetry key pair. Then, the JWT Session Token, UUID v4 and Public Key will be sent to the Console for device registration.

The TOE also allow the user to reset their password, setup and reset the Two-Factor Authentication of the TOE through the Account Interface. In addition, the TOE also receive login history of the user which includes the device used by the TOE user, location, the date and time of last accessed and the operations of the TOE user on the TOE from the Console and listed in the Account Interface.

The General Information Interface outlines the Name, Account and the Domain of the TOE user, also the TOE user will be able to access the Account Interface and logout of the TOE from this interface.

To establish a connection, the user have to first be authenticated on the TOE. After the user is authenticated, they can send a request to establish a connection with their targeted backend resources through the Tunnel Establishment Interface. This information will be sent to the Console, and once the request is verified, the console will obtain and send the connection information back to the TOE and the ServiceHostAgent (i.e., backend resources) to establish a tunnel connection. After the tunnel is established, the user will be able to access the backend resources remotely and securely.

Lastly, the TOE allow the user to terminate their own interactive session through the “logout” button presented in the General Information Interface and Account interface, or through the “Terminate” button presented under the “Operations” column, under “Login History” found within the Account Interface.

1.7.1 PHYSICAL SCOPE OF TOE

The TOE is a next-generation Virtual Private Network (VPN), a software application which provides end-to-end secure tunnelling channels that are resistant to quantum attacks from the user's device to respective backend resources.

The non-TOE comprise of the following:

- **Transport Layer Security.** Cryptography of the communication channel between users to the TOE; and from TOE to backend resources with TLS version 1.2 and 1.3, which is provided by:
  a) https://pkg.go.dev/crypto/tls (go1.20.5)
  b) https://boringssl.googlesource.com/boringssl
- **Secret Code Authentication Scheme**. Two-Factor Authentication that provides secret code by external authentication schemes, which includes:
  a) Time-Based One Time Password (TOTP) from preferred authenticator application.
  b) EZAuth which is an authenticator application developed by Chelpis that provides additional authentication method through a separate user device.

  NOTE: The evaluation of the 2FA TOE component is performed on G2FA for TOTP and EZAuth Authenticator for EZAuth.
- **ServiceHostAgent.** Also known as Endpoint or Host, which are endpoints that the user wants to communicate with remotely The agent is installed on ServiceHostAgent/device to be activated and allow tunnel establishment to occur.
- **ServiceHostAgent/device.** This represents the underlying platform (e.g., computer, server, databases, etc.) of the Endpoint or Host. After an administrator authenticates with the Console using a ServiceHostAgent, the computer is registered to the device list of that service server.
- **Console.** The console is a software for VPN management which provides management functions such as access control policy, audit logs, host managements, and so on. Console is managed by administrator.
- **Platform (i.e., UserAgent/device) for the TOE.** It represents any system (e.g., computer) the user logs into. After a user authenticates with the Console through the TOE on any computer, the computer is registered to the user's device list. The underlying platform for the TOE is as below:
  a) Windows 10 (21H1) and above (x64/ arm64); or
  b) MacOS Big Sur 11 and above (x64/ arm64).

  NOTE: The evaluation of the TOE component is performed on MacOS version 14.1.1, arm64.
- **Device Interface.** The device interface contains information related to the User's device, such as operating system and the virtual IP. This interface also outlines the TOE version, user device's public key and local event logs, which are not included in this evaluation.

1.7.2 TOE GUIDANCE

The TOE guidance documents comprise of the following:

- PQTunnel Installation Manual v1.0.2
- PQ-003 Chelpis PQTunnel v1.1.5 Preparative Guidance v1.9
- PQ-004 Chelpis PQTunnel v1.1.5 Operational Guidance v1.9

1.7.3 LOGICAL SCOPE OF TOE

The logical scope of TOE is described based on several security functional requirements.

1.7.3.1 IDENTIFICATION AND AUTHENTICATION

The TOE users are required to provide their Account, Password and Domain before able to gain access to the TOE and initiate secure connections to backend resources. The TOE also require the user to utilise credentials with certain level of quality, and the login attempts are limited to 20 tries within 10 minutes.

1.7.3.2 CRYPTOGRAPHIC SUPPORT

The TOE provide the full lifecycle of cryptographic key generation, key destruction, and key operation for different operations for establishing and maintaining secure tunnel channels.

1.7.3.3 TRUSTED PATH/ CHANNELS

The TOE provide secure path and channels for data in-transit during communications between local user and the TOE; and also between the TOE and other trusted IT entities. This security is to prevent unauthorised disclosures and modifications for the data in-transit.

1.7.3.4 SECURITY MANAGEMENT

The TOE provide security management capability for the TOE user to reset their password, setup and reset the Two-Factor Authentication of the TOE through the Account Interface.

1.7.3.5 SECURITY AUDIT

The TOE allows the TOE user to view their login history including the device used by the TOE user, location, the date and time of last accessed and the operations of the TOE user on the TOE and shown through the Account Interface. These information are obtained from the Console.

1.7.3.6 TOE ACCESS

The TOE allow the user to terminate their own interactive session. This is performed by the user's own action through pressing the "logout" button through the General Information Interface or Account Interface, or by pressing the "terminate" button through the Account Interface, under the "Operations" column.

2 CONFORMANCE CLAIMS

2.1 COMMON CRITERIA CONFORMANCE CLAIM

The ST and the TOE described are claimed as followed:

- Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, April 2017 **conformant**.
- CC part 2 **conformant**.
- CC part 3 **conformant**.

2.2 ASSURANCE PACKAGE CLAIM

This Security Target claims conformance to Evaluation Assurance Level 1.

2.3 PROTECTION PROFILE CONFORMANCE CLAIM

This ST does not claim conformance of the TOE with any Protection Profile (PP).

3 SECURITY OBJECTIVES

The purpose of the security objectives refers to high-level statements or goals that describe the desired security outcome for the TOE. Security objectives are defined to address the protection needs of the TOE and its operational environment. The security objective on the operational environment is to address the security concerns of the conditions and characteristics in which a TOE is intended to operate.

3.1 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

This section identifies and describes the security objectives that are to be addressed by the IT environment or by non-technical or procedural means.

Table 3: Operational Environment's Security Objectives

Security Objective	Description
OE.PLATFORM	The TOE relies upon a trustworthy computing platform. This includes the underlying platforms provided to the TOE.
OE.USER	The TOE user of the TOE is not wilfully negligent or hostile and uses the software in compliance with the applied security policy and guidance. TOE user must be aware and do not leave their device unattended without screen lock mechanism.
OE.2FA_KEY	The 2FA for identification and authentication is provided by a separate IT product (e.g., hardware, authenticator, etc.), that particular product has been evaluated to be secure and do not allow tampering or bypass.
OE.PHYSICAL	The TOE is installed on devices that are accessible only to authorised personnel; and the endpoints are located in secure areas accessible only to authorised personnel.
OE.RBG	Random bits generation is a critical component in cryptographic key generation and other cryptographic operations. In the context of complying with the FIPS SP 800-90A Rev 1 standard, the randomness used in these operations should adhere to the requirements specified in the standard. This is dependent on the underlying platform.
OE.RNG	Random number generator is a critical component in generation of UUID v4. In the context of complying with the RFC 4122 standard, the randomness used in these operations should adhere to the requirements specified in the standard. This is dependent on the underlying platform.
OE.TIMESTAMP	Time stamp is important for the Audit Log generated to have reliable point of reference. The platform where the TOE is installed shall have a reliable time source.

4 EXTENDED COMPONENTS DEFINITION

4.1 SECURITY OBJECTIVES REQUIREMENTS

This ST does not include extended Security Functional Requirements.

4.2 SECURITY ASSURANCE REQUIREMENTS

This ST does not include extended Security Assurance Requirements.

5 SECURITY REQUIREMENTS

This section provides security functional and assurance requirements that must be satisfied by a compliant TOE. These requirements consist of functional components from Part 2 of the CC, and an Evaluation Assurance Level (EAL) that contains assurance components from Part 3 of the CC.

5.1 CONVENTIONS

The CC permits four types of operations to be performed on functional requirements: selection, assignment, refinement, and iteration. These operations, when performed on requirements that derive from CC Part 2, are identified in this ST in the following manner:

- Selection: Indicated by surrounding brackets, e.g., [selected item].
- Assignment: Indicated by surrounding brackets and italics, e.g., [*assigned item*].
- Refinement: Refined components are identified by using **bold** for additional information, or ~~strikeout~~ for deleted text.
- Iteration: Indicated by assigning a number in parenthesis to the end of the functional component identifier as well as by modifying the functional component title to distinguish between iterations, e.g., 'FDP_ACC.1(1), Subset access control (authorised admins)' and 'FDP_ACC.1(2) Subset access control (devices)'.

5.2 TOE SECURITY FUNCTIONAL REQUIREMENTS

The security functional requirements for this ST consist of the following components from Part 2 of the CC and extended components defined in Section 4, summarized in **Table *4*** - Summary of Security Functional Requirements.

Table 4: Summary of Security Functional Requirements

Class	Identifier	Name
Identification and Authentication (FIA)	FIA_ATD.1	User attribute definition
	FIA_UID.2	User identification before any action
	FIA_UAU.2	User authentication before any action
	FIA_SOS.1	Verification of secrets
	FIA_AFL.1	Authentication failure handling
Cryptographic Support (FCS)	FCS_CKM.1(1)	Cryptographic key generation (Authentication)
	FCS_CKM.1(2)	Cryptographic key generation (PQC Shared Secret)
	FCS_CKM.1(3)	Cryptographic key generation (Shared Secret)
	FCS_CKM.1(4)	Cryptographic key generation (AES keys)
	FCS_CKM.1(5)	Cryptographic key generation (UUID v4)
	FCS_CKM.4	Cryptographic key destruction
	FCS_COP.1(1)	Cryptographic operation (Authentication)
	FCS_COP.1(2)	Cryptographic operation (PQC-KEM)
	FCS_COP.1(3)	Cryptographic operation (Diffie-Hellman)
	FCS_COP.1(4)	Cryptographic operation (AEAD)
Trusted Path/ Channels (FTP)	FTP_TRP.1	Trusted path
	FTP_ITC.1	Inter-TSF trusted channel
Security Management (FMT)	FMT_SMR.1	Security roles
	FMT_SMF.1	Specification of management functions
Security Audit (FAU)	FAU_SAR.1	Audit Review
TOE Access (FTA)	FTA_SSL.4	User-initiated termination

5.2.1 IDENTIFICATION AND AUTHENTICATION (FIA)

5.2.1.1 FIA_ATD.1: User attribute definition

Hierarchical to: No other components.
Dependencies: No dependencies.

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: [

a) Account
b) Password
c) Domain].

5.2.1.2 FIA_UID.2: User identification before any action

Hierarchical to: FIA_UID.1 Timing of identification
Dependencies: No dependencies.

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

5.2.1.3 FIA_UAU.2: User authentication before any action

Hierarchical to: FIA_UAU.1 Timing of authentication
Dependencies: FIA_UID.1 Timing of identification

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

5.2.1.4 FIA_SOS.1: Verification of secrets

Hierarchical to: No other components.
Dependencies: No dependencies.

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that ~~secrets~~ **passwords** meet: [*Minimum of 12 and maximum of 32 characters*].

5.2.1.5 FIA_AFL.1: Authentication failure handling

Hierarchical to: No other components.
Dependencies: FIA_UAU.1 Timing of authentication

FIA_AFL.1.1 The TSF shall detect when [positive integer number:[*20*]] unsuccessful authentication attempts occur related to [*authentication attempts to the TOE*].

FIA_AFL.1.2 When the defined numbers of unsuccessful authentication attempts has been [surpassed], the TSF shall [*disable the user account*].

APPLICATION NOTE: The time duration for the unsuccessful authentication attempts to the TOE is 10 minutes.

5.2.2 CRYPTOGRAPHIC SUPPORT (FCS)

5.2.2.1 FCS_CKM.1(1): Cryptographic key generation (Authentication)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1 The TSF shall generate **asymmetric** cryptographic keys **used for authentication (Let Console verify the authenticity of a Device)** in accordance with a specified cryptographic key generation algorithm: [

- *Curve P-256*
- *ECC Key Pair Generation*]

and specified cryptographic key sizes [*32 bytes*] that meet the following: [*FIPS PUB 186-4, Appendix B.4*]

5.2.2.2 FCS_CKM.1(2): Cryptographic key generation (PQC Shared Secret)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1 The TSF shall generate **asymmetric post-quantum** cryptographic keys **used for letting two parties establish a shared secret that is resistant to cryptanalysis by future adversaries with quantum computers** in accordance with a specified cryptographic key generation algorithm: [

- *CRYSTALS-Kyber 768*]

and specified cryptographic key sizes [*2,400 bytes*] that meet the following: [*Post-Quantum Cryptography (CRYSTALS-KYBER) (https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions)*]

5.2.2.3 FCS_CKM.1(3): Cryptographic key generation (Shared Secret)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1 The TSF shall generate **asymmetric** cryptographic keys **used for letting two parties establish a shared secret** in accordance with a specified cryptographic key generation algorithm: [*Curve25519*] and specified cryptographic key sizes [*32 bytes*] that meet the following: [*RFC 7748 https://datatracker.ietf.org/doc/html/rfc7748*]

5.2.2.4 FCS_CKM.1(4): Cryptographic key generation (AES keys)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1 The TSF shall generate ~~cryptographic keys~~ **AES keys** in accordance with a specified cryptographic key generation algorithm: [

- *BLAKE2s*
- *HKDF*]

and specified cryptographic key sizes [*32 bytes*] that meet the following: [

- *RFC 7693 https://datatracker.ietf.org/doc/html/rfc7693*
- *RFC 5869 https://datatracker.ietf.org/doc/html/rfc5869*]

5.2.2.5 FCS_CKM.1(5): Cryptographic key generation (UUID v4)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1 The TSF shall generate ~~cryptographic keys~~ **Universally Unique Identifier** in accordance with a specified cryptographic key generation algorithm [*random number generator]*and specified cryptographic key sizes [*16 bytes*] that meet the following: [*RFC 4122*].

APPLICATION NOTE: FCS_CKM.2 and FCS_COP.1 are both not applicable due to the UUID v4 is not generated for cryptography operation, but to bind a user device with unique identifier.

5.2.2.6 FCS_CKM.4: Cryptographic key destruction

Hierarchical to: No other components.
Dependencies: [FDP_ITC.1 Import of user data without security attributes,

or FDP_ITC.2 Import of user data with security attributes,
or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [*single overwrite consisting of zeroes*] that meets the following: [*no standard*].

5.2.2.7 FCS_COP.1(1) Cryptographic operation (Authentication)

Hierarchical to: No other components.
Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]
FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1 The TSF shall perform [*Digital Signature used for authentication (Let "Control Center" verify the authenticity of a "Device")*] in accordance with a specified cryptographic algorithm: [

- *Curve P-256*
- *ECDSA*]

and cryptographic key sizes [*32 bytes*] that meet the following: [*FIPS PUB 186-4, Section 6.4, ECDSA Digital Signature Generation and Verification*]

5.2.2.8 FCS_COP.1(2) Cryptographic operation (PQC-KEM)

Hierarchical to: No other components.
Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]
FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1 The TSF shall perform [*encapsulation mechanism*] in accordance with a specified cryptographic algorithm [*Kyber-768*] and cryptographic key sizes [*32 bytes*] that meet the following: [*Post-Quantum Cryptography (CRYSTALS-KYBER) (https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions)*]

5.2.2.9 FCS_COP.1(3) Cryptographic operation (Diffie-Hellman)

Hierarchical to: No other components.
Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]
FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1 The TSF shall perform [*Diffie-Hellman*] in accordance with a specified cryptographic algorithm [*X25519*] and cryptographic key sizes [*32 bytes*] that meet the following: [*RFC 7748 (https://datatracker.ietf.org/doc/html/rfc7748)*]

5.2.2.10 FCS_COP.1(4) Cryptographic operation (AEAD)

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]
FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1 The TSF shall perform [*AEAD*] in accordance with a specified cryptographic algorithm [*ChaCha20Poly1305*] and cryptographic key sizes [*32 bytes*] that meet the following: [*RFC 8439 https://datatracker.ietf.org/doc/html/rfc8439*]

5.2.3 TRUSTED PATH/ CHANNELS (FTP)

5.2.3.1 FTP_TRP.1: Trusted Path

Hierarchical to: No other components.
Dependencies: No dependencies.

FTP_TRP.1.1 The TSF shall provide a communication path between itself and [local] users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [modification, disclosure].

FTP_TRP.1.2 The TSF shall permit [local users] to initiate communication via the trusted path.

FTP_TRP.1.3 The TSF shall require the use of the trusted path for [initial user authentication].

5.2.3.2 FTP_ITC.1: Inter-TSF trusted channel

Hierarchical to: No other components.
Dependencies: No dependencies.

FTP_ITC.1.1 The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2 The TSF shall permit [the TSF, another trusted IT product] to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for [*sending connection establishment request to the Console, receiving tunnel connection information from the Console, establishing tunnel with the Agent*].

5.2.4 SECURITY MANAGEMENT (FMT)

5.2.4.1 FMT_SMR.1: Security roles

Hierarchical to: No other components.
Dependencies: FIA_UID.1 Timing of identification

FMT_SMR.1.1 The TSF shall maintain the roles [*User*].

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

5.2.4.2 FMT_SMF.1: Specification of management functions

Hierarchical to: No other components.
Dependencies: No dependencies.

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions **for password**: [*Reset*]**; and for 2FA:** [*Setup, Reset*].

5.2.5 SECURITY AUDIT (FAU)

5.2.5.1 FAU_SAR.1: Audit review

Hierarchical to: No other components.
Dependencies: FAU_GEN.1 Audit data generation

FAU_SAR.1.1 The TSF shall provide [*User*] with the capability to read [*Login History (Device, Location, Last Accessed, Operations)*] from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

APPLICATION NOTE: FAU_GEN.1 is not applicable due to the Audit Logs are not generated by the TOE, but by the Console which is a non-TOE component. The TOE only receive the Audit Log from the Console and present it to the TOE user.

5.2.6 TOE ACCESS (FTA)

5.2.6.1 FTA_SSL.4: User initiated termination

Hierarchical to: No other components.
Dependencies: No dependencies.

FTA_SSL.4.1 The TSF shall allow user-initiated termination of the user's own interactive session.

5.3 DEPENDENCY RATIONALE

Table *5* identifies the Security Functional Requirements from Part 2 of the CC and their associated dependencies. It also indicates whether the ST explicitly addresses each dependency.

Table 5: Security Functional requirement dependencies

SFR	Dependency	Dependency Satisfied	Rationale
FIA_ATD.1	None	-	
FIA_UID.2	None	-	
FIA_UAU.2	FIA_UID.1	✓	The dependency is met by FIA_UID.2, which is hierarchical to FIA_UID.1
FIA_SOS.1	None	-	
FIA_AFL.1	FIA_UAU.1	✓	The dependency is met by FIA_UAU.2, which is hierarchical to FIA_UAU.1
FCS_CKM.1(1)	FCS_CKM.2 OR FCS_COP.1	✓	The dependency is met by FCS_COP.1(1)
	FCS_CKM.4	✓	
FCS_CKM.1(2)	FCS_CKM.2 OR FCS_COP.1	✓	The dependency is met by FCS_COP.1(2)
	FCS_CKM.4	✓	
FCS_CKM.1(3)	FCS_CKM.2 OR FCS_COP.1	✓	The dependency is met by FCS_COP.1(3)
	FCS_CKM.4	✓	
FCS_CKM.1(4)	FCS_CKM.2 OR FCS_COP.1	✓	The dependency is met by FCS_COP.1(4)
	FCS_CKM.4	✓	
FCS_CKM.1(5)	FCS_CKM.2 OR FCS_COP.1	X	Not applicable as the UUID v4 is not distributed nor used as cryptographic key pair, it is to associate a device for identifier.
	FCS_CKM.4	✓	
FCS_CKM.4	FDP_ITC.1 OR FDP_ITC.2 OR FCS_CKM.1	✓	The dependency is met by FCS_CKM.1
FCS_COP.1(1)	FDP_ITC.1 OR FDP_ITC.2 OR FCS_CKM.1	✓	The dependency is met by FCS_CKM.1(1)
	FCS_CKM.4	✓	

FCS_COP.1(2)	FDP_ITC.1 OR FDP_ITC.2 OR FCS_CKM.1	✓	The dependency is met by FCS_CKM.1(2)
	FCS_CKM.4	✓	
FCS_COP.1(3)	FDP_ITC.1 OR FDP_ITC.2 OR FCS_CKM.1	✓	The dependency is met by FCS_CKM.1(3)
	FCS_CKM.4	✓	
FCS_COP.1(4)	FDP_ITC.1 OR FDP_ITC.2 OR FCS_CKM.1	✓	The dependency is met by FCS_CKM.1(4)
	FCS_CKM.4	✓	
FTP_TRP.1	None	-	
FTP_ITC.1	None	-	
FMT_SMR.1	FIA_UID.1	✓	The dependency is met by FIA_UID.2, which is hierarchical to FIA_UID.1
FMT_SMF.1	None	-	
FAU_SAR.1	FAU_GEN.1	X	Not applicable as the Audit Log is not generated by the TOE, but by the Console which is a non-TOE component.
FTA_SSL.4	None	-	

5.4 TOE SECURITY ASSURANCE REQUIREMENT

The TOE assurance requirements for this ST consist of the requirements corresponding to the EAL 1 level of assurance, as defined in the CC Part 3. This EAL was chosen based on the security problem definition and the security objectives for the TOE. The chosen assurance level is consistent with the claimed threat environment.

The assurance requirements are summarized in **Table 6** – Security Assurance Requirements.

Table 6: Security Assurance Requirements (EAL1)

Assurance Class	Assurance Components
ADV: Development	ADV_FSP.1 Basic functional specification
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.1 Labelling of the TOE
	ALC_CMS.1 TOE CM coverage
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	ASE_INT.1 ST introduction

	ASE_OBJ.1 Security objectives for the operational environment
	ASE_REQ.1 Stated security requirements
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_IND.1 Independent testing - conformance
AVA: Vulnerability assessment	AVA_VAN.1 Vulnerability survey

6 TOE SUMMARY SPECIFICATION

This section provides a description of the security functions and assurance measures of the TOE that meet the TOE security requirements.

6.1 IDENTIFICATION AND AUTHENTICATION

The users must be identified and authenticated before granted access to any functionality of the TOE. This is to protect the TOE from illegitimate access and outside threats. This security functions is provided by FIA_UID.2 and FIA_UAU.2.

The users are required to provide the security attributes which are Account, Password and Domain to successfully initiate communication with the TOE. This security function is provided by FIA_ATD.1.

The password provided by the users are required to meet certain quality metrics, where they must be a minimum of 12 and maximum of 32 characters. This quality metric is not applicable for Account and Domain. This security function is provided by FIA_SOS.1.

To prevent common attacks such as credential stuffing, the TOE is able to detect the number of attempts the user tried to login. When there are more than 20 unsuccessful authentication attempt within 10 minutes, the user account will be disabled by the TSF. This security function is provided by FIA_AFL.1.

TOE Security Functional Requirements addressed: FIA_UID.2, FIA_ATD.1, FIA_UAU.2, FIA_SOS.1, FIA_AFL.1

6.2 CRYPTOGRAPHIC SUPPORT

The TOE implements:

- FCS_CKM.1 Cryptographic Keys Generation for Authentication, UUID v4, PQC Shared Secret, Shared Secret and AES keys.
- FCS_CKM.4 Cryptographic Keys Destruction through single overwrite consisting of zeroes.
- FCS_COP.1 Cryptographic Operation to ensure the cryptographic operations are performed according to specified algorithms, key sizes and standards for Authentication, PQC-KEM, Diffie-Hellman and AEAD.

Please refer to **Table 7** until **Table 11** for further explanations on the cryptographic generation, operation and destruction for each processes.

Table 7: The cryptographic generation, operation and destruction for Authentication.

Question	Description
Why is this needed?	To allow Console to verify the authenticity of a Device used by the user.
When will this be generated?	• The first time user login on a new device, which is after user authenticated by Console. • Every time user re-login (not the first time) or re-launch the TOE, it will generate new key and replace the old one.
How is it used?	After it is generated, it will be registered to the Console with the user's JWT token, which was provided to the user after successful authentication of their credentials (i.e., Account, Password and Domain), along with the UUID v4 of the device. The device's interaction with the Console requires a signature authentication via the use of an Authentication key during API calls. It's important that the Console authenticates this signature before deciding to carry out the operation. The secure and trusted interaction between our device and the Console ensures our system's efficiency and data integrity.

When and how is it destroyed?	• Single overwrite consisting of zeroes when:
	  ○ Every time user re-login (not the first time) or re-launch the TOE, it will be rotated.
	  ○ User delete the TOE.

Table 8: The cryptographic generation, operation and destruction for UUID v4.

Question	Description
Why is this needed?	UUID v4 is required to identify the device used by the user for improved security.
When will this be generated?	The first time user login on a new device, which is after user authenticated by Console.
How is it used?	UUID v4 will be generated using a random number generator and will be used as a unique identifier for the device used by the user. The UUID v4 will be sent to the Console along with the JWT session token and Public Key.
When and how is it destroyed?	• Single overwrite consisting of zeroes when: ○ User delete the TOE.

Table 9: The cryptographic generation, operation and destruction for PQC-KEM.

Question	Description
Why is this needed?	The PQC-KEM (Post-Quantum Cryptography Key Encapsulation Mechanism) is required to ensure that the keys of AEAD possess quantum resistance. Originally, the AEAD algorithm obtains a shared secret through key exchange via X25519, which unfortunately does not offer quantum resistance. PQC-KEM, through its key exchange mechanism based on post-quantum cryptography, guarantees quantum resistance and thereby provides quantum security for the communication channel.
When will this be generated?	The PQC-KEM is generated when a user attempts to establish a secure channel through tunnel establishment.

How is it used?	i. Endpoint 1 (i.e., user device) generates an asymmetric key pair (public key 1 & private key 1) through the keygen function. ii. Endpoint 1transmits the public key 1 to Endpoint 2 (i.e., backend resource) via a trusted signalling channel. iii. Endpoint 2 randomly generates plaintext, feeds the plaintext and public key 1 into the encapsulate function, and obtains ciphertext. This ciphertext is transmitted to Endpoint 1 via the trusted signalling channel. iv. Upon receiving the ciphertext, Endpoint 1 inputs the ciphertext and private key 1 into the decapsulate function to retrieve the plaintext. v. At this point, both sides have completed key exchange through quantum-resistant KEM.
When and how is it destroyed?	The PQC-KEM is immediately destroyed once the two endpoints exchange the shared secret through single overwrite consisting of zeroes.

Table 10: The cryptographic generation, operation and destruction for Diffie-Hellman.

Question	Description
Why is this needed?	Diffie-Hellman is needed to allow two parties to establish a shared secret, enhancing the security of their communication.
When will this be generated?	The asymmetric keys based on Curve25519 are generated as soon as the user login to the TOE.
How is it used?	To establish a connection, the two endpoints each generate asymmetric keys based on Curve25519 after user login. Through Diffie-Hellman, they perform key exchange to obtain a shared secret.
When and how is it destroyed?	The keys are deleted through single overwrite consisting of zeroes when the user closes the TOE or when the program encounters anomalies, such as being unable to establish a connection with the Console for a certain period of time.

Table 11: The cryptographic generation, operation and destruction for AEAD.

Question	Description
Why is this needed?	AEAD is required for symmetric encryption, providing data protection within the channel.
When will this be generated?	Following the completion of Diffie-Hellman & PQC-KEM, the shared secret obtained from these processes is used to generate AES keys via BLAKE2s and HKDF (HMAC-based Key Derivation Function).
How is it used?	AEAD is utilized as a cipher mode that provides authenticated encryption with associated data through ChaCha20Poly1305 algorithm.
When and how is it destroyed?	AES keys are rotated periodically to maintain security. It is promptly destroyed by single overwrite consisting of zeroes once the channel is closed.

TOE Security Functional Requirements addressed: FCS_CKM.1(1,2,3,4,5), FCS_CKM.4, FCS_COP.1(1,2,3,4)

6.3 TRUSTED PATH/ CHANNELS

The TOE ensures that the communications between the local user to the TOE; and TOE to Console and Agents. This is employed with the utilisation of cryptography of the communication channels.

The communication channel between the local user and TOE is protected from inadvertent modification and disclosure whenever there are initial user authentication. This security function is provided by FTP_TRP.1.

For communications between the TOE with other trusted IT products, the communication channel is also protected from unauthorised modification and disclosure. This security function is enforced when the TOE sends connection establishment request to the Console, receives tunnel connection information from the Console, and establishes tunnel with the Agent. This security function is provided by FTP_ITC.1.

TOE Security Functional Requirements addressed: FTP_TRP.1, FTP_ITC.1

6.4 SECURITY MANAGEMENT

The TOE provides specific role for the TOE user when they are operating the TOE. The role assigned to the TOE user is “User”, which can be verified in the Account Interface. This security function is provided by FMT_SMR.1.

There are a few security management that can be performed by the TOE user, they are able to reset their password, setup their 2FA and reset their 2FA. These security management functions are also accessible through the Account Interface. This security function is provided by FMT_SMF.1.

TOE Security Functional Requirements addressed: FMT_SMR.1, FMT_SMF.1

6.5 SECURITY AUDIT

The TOE user is able to view their login history using the TOE. The TOE receives the Audit Logs from the Console, which are the device used by the TOE user, location, the date and time of last accessed and the operations of the TOE user on the TOE and shown through the Account Interface. This security function is provided by FAU_SAR.1.

TOE Security Functional Requirements addressed: FAU_SAR.1

6.6 TOE ACCESS

Once the TOE user has completed their operation on the TOE, they are able to initiate the termination of their own interactive session through two methods. First method is by pressing the “Logout” button in the General Information Interface or Account Interface; while the second method is by pressing the “Terminate” button in the “Operations” column of “Login History”, which can be found in the Account Interface. This security function is provided by FTA_SSL.4.

After logging out, the user will be required to authenticate themselves by providing their Account, Password and Domain again before they are able to access the TSF and TOE data.

TOE Security Functional Requirements addressed: FTA_SSL.4