SECURITY TARGET

Clarabot Nano Client

Contents

List of Figures

List of Tables

1. ST Introduction

1.1. ST reference

ST reference: Clarabot Nano Security Target (CNC - ST)

ST version: 0.16

ST date: 2023-02-03

CC version: ISO/IEC 15408:2022

Assurance level: EAL2

ST author: Clarabot Zrt.

1.2. TOE reference

The TOE reference is "Clarabot Nano Client version 1.1.0" (or Nano Client or CNC) (in the evaluated configuration).

1.3.TOE overview

The Clarabot Nano system provides a data sharing platform for its users.

There are two types of traditional network models for data sharing:

- Peer-to-Peer model: where all computers are connected with each other, and
- Client-Server model: where all computers are connected to a server computer that facilitates the file sharing.

Nano’s network is a hybrid server-client network.

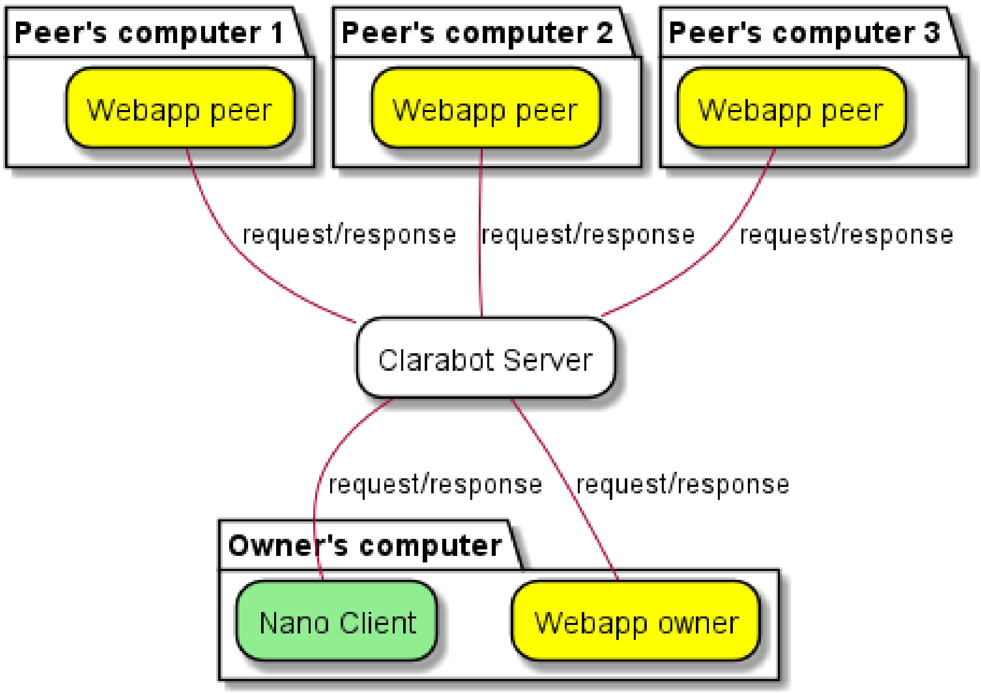

Figure 1 The Clarabot Nano system

The Clarabot Nano software pair (Nano Client and Nano "Webapp owner" in Figure 1.1) facilitates live and secure file sharing. When Nano Client is installed on a user's computer, it will provide controlled remote access to its resources. The permissions and properties of remotely accessible resources are exclusively determined by the Owner of the account the Nano Client is using. All user content is secured by end-to-end authenticated encryption between the Nano Client and Nano web applications.

The Nano system has centralized Nano Clarabot Servers (Clarabot Server in Figure 1.1) which act as proxy for the data sharing network. All requests/responses between a Nano user and another Nano user are routed through the Clarabot Servers. A Clarabot Server hides the IPs of the machines running Nano Client and facilitates secure communication between Nano users, and by extension, group of users. Without the right permissions, no Nano user is able to snoop into the files of a different Nano user. The hybrid server-client network also makes it possible to cache the separated, encrypted data streams for a shorter request round-trip-time.

The Nano web application (Webapp Owner and Webapp peer in Figure 1) is served from the Clarabot Server directly. The Nano user's account is managed by the web application communicating with the Clarabot Server.

The TOE is the Nano Client (see the area tinted green in the Figure 1) int the evaluated configuration (where explicit trust for new peers is compulsory).

1.3.1. TOE type

The Clarabot Nano Client is a secure file-sharing software application.

This application enables and equips effective collaboration and file sharing with persons specially authorised to do so. It is a multi-process application running in the background without interrupting the operating system. It is responsible for handling and executing all file-sharing and indexing operations on the local machine.

The Nano Client is also the main security entry point for authorizing all remote file access requests.

1.3.2. TOE usage

Registration

In order to connect to the data sharing platform (resource sharing network), future Nano users must have an account registered. The registration process can be carried out through the Nano web application (i.e. outside the scope of the TOE).

After the account is created, the Nano Client will construct a keyring (that will facilitate the cryptography) and encrypts it in a way that only by knowing the account's password it is possible to decrypt it. The encrypted Account Keyring is stored on the server for the account.

Enforced trust (explicit trust for peers)

The evaluated TOE is the Nano Client with "explicit trust option is ON "setting. In this case the TOE does not accept requests without prior identity trust establishment between the entities (Owner and Peer).

Drives and Rooms

The first step in sharing files through the Nano Client is to designate a **Drive**.

A Drive is a folder on which the Nano Client has permission to do its resource sharing capabilities and other supported operations.

When a Drive is set up, the Nano Client will parse and index each file and folder within the Drive folder. Building on these preparations, the Nano Client can do different kind of operations on the assigned Drive like listing, searching, creating, deleting, moving, copying or updating files or folders.

The Drive folder acts as a container for file system related operations. Any sort of operation outside this Drive folder is not permitted for the Nano Client. The Nano Client must not and will not know about anything outside the Drive's root folder. Any type of references (shortcuts, symlinks etc.) pointing outside the Drive folder structure won't work; they'll appear broken.

Rooms are entities on the server that the Owner creates through the Nano web application.

The Nano users can freely create new Rooms on the fly for different kind of purposes. They can invite other Nano users to their Room, allowing them access to the contents within.

A Nano Client can be bound to a Room using a Drive.

Rooms have a configuration stored by the server that contains the membership and group permissions is secured by the owner's digital signature and block-chain properties.

A Room-key is generated at Room creation by the Owner. This key will be shared on a peer-to-peer basis with each account that is invited to the Room.

Search, browse, request and response

The Nano users can browse and perform searches within all Rooms owned by them, and all Rooms which are shared to them by other Nano users.

Searching and browsing are done in the Nano web application, based on the drives attached to the owned Rooms.

The search results will contain detailed information about each result found and also the reason why it satisfies the search terms.

When searching for files, the search is done in the file name and their content as well.

Based on search results a Nano user (from a peer's computer in Figure 1) can request different types of remote operations from Drives assigned to the Room. If the requestor has the right to do so, the request will be executed and the result will be included in a response (from the Owner's computer in Figure 1).

Cacheable response

To improve performance and increase the efficiency of data sharing the Nano system also supports cacheable responses. Such acceleratory, resource-saving responses provide the following:

- The Clarabot Server is able to store the responses a relevant fast-access cache storage
- The Clarabot Server does not respond from cache without the consent of the TOE (recipient of the request)
- Responding to requests from this cache storage is as secure as responding from the TOE.

1.3.3. Major security features of the TOE

The security features of the Clarabot Nano Client include but is not limited to:

Zero knowledge

Zero knowledge means that apart from the Owner and the legitimate Peer no other Nano user can access the transmitted data. Not even the Clarabot Server have the ability to decrypt the transmitted data.

End-to-end encryption

End-to-end encryption means that no sensitive data leaves the Owner’s local machine in clear form. (Some limited data of a user account is not encrypted, for example the user’s email address. This data is required to be kept unencrypted in order for the Clarabot Server to provide certain services.)

Everything goes through cipher processes before it’s handed over to the messaging pipeline, only to be decrypted by the receiving endpoint. Because of this process, there is no way to maliciously alter the data being sent without the receiving end noticing.

The end-to-end encryption combined with the zero knowledge design means that no other parties (not even the Clarabot Server) are able to access the user’s data.

Layered Cryptography

At the time of registration, multiple keys are set-up for the Owner for different purposes. These keys are stored in encrypted form, and it is only possible to decrypt them after entering a password. It’s crucial to choose a strong password for the user account.

Using a strong password allows the TOE to set up a durable security fortress around the Owner’s private data. The security implementation consists of multiple layers of different cryptographic methods. This ensures the security of the shared data within Nano's file sharing system.

1.3.4.Required non-TOE hardware/software/firmware

The following hardware, firmware and software supplied by the IT environment are excluded from the TOE boundary (see Figure 1):

System side:

- Clarabot Server

Client side:

- Hardware (of the Owner’s computer and Peer’s computer)
- OS (of the Owner’s computer and Peer’s computer)
- Nano webapp (on the Owner’s computer and Peer’s computer)
- CSPRNG (a Cryptographically Secure Pseudorandom Number Generator)
- text viewer application

1.4.TOE description

The generic architecture of the CNC is shown in (Figure 1).

1.4.1.The physical scope of the TOE

The TOE is available in prebuilt binary distribution packages for the following major platforms:

- Nano for Windows 10 or later (x64) /clarabot-nano-1.1.0-win-amd64.exe/
- Nano for macOS 11 or later (Apple Silicon) /clarabot-nano-1.1.0-aarch64.dmg/
- Nano for macOS 10.15 or later (Intel) / clarabot-nano-1.1.0-intel64.dmg/
- Linux AppImage for glibc 2.17 or later (kernel 3.10) /clarabot-nano-1.1.0-amd64.AppImage/

The binary packages are downloadable from website: https://www.clarabot.com/download

The shared source code of the different platforms is written in Python.
The prebuilt binary images contain the required dependencies for running the application.

The operational and installation guides are also available from website:

- https://www.clarabot.info/nano/en/client-guide/
- https://www.clarabot.info/nano/en/client-guide/installation/

1.4.2.The logical scope of the TOE

This section lists the logical scope of the TOE, including the major TOE functions and provide a brief description of the security features (the TSF).

Identification and authentication

The TOE associates roles to users by identity and authorization by context. A requestor may have the identity of Owner, Peer or Anonymous. A Peer may get various authorizations by the context of a Room, like RoomAdministrator.

A logon mechanism assumes that the Owner already registered an account in Clarabot Server. Identification and authorization is a cryptographically secured multi-step process. It involves the TOE creating a Private Keyring from the Owner's email address and their password. The keys in the keyring are either used as authentication keys presented to the server or used for encryption of other keys that the server stores. The Private Keyring is always deleted after use during the logon process.

The TOE only accepts requests in authenticated-encryption, thus, it is able to identify the requestor's role, and even its account id in the case of the Peer and RoomAdministrator.

Keyring import

A symmetric master key is generated for the account firstly during registration using CSPRNG. It is encrypted by a derived secret key in the Private Keyring from the login, so only the Owner may decipher it.

During account creation two asymmetric key-pair are generated: an Ed25519 for signing operation and a Curve25519 for key exchange.

The secret parts of these key-pairs are encrypted using the master key, that only their owner can decrypt using their Private Keyring.

The Owner's Account Keyring is always encrypted in storage, it includes the symmetric secret keys and asymmetric secret key-parts.

The public parts of the asymmetric keys are included in the Peer's public keyring.

Both types of keyrings are stored on the Clarabot Server (the secret one only in encrypted form).

The TOE imports the Owner's (secret) keyring during the logon process after the successful authentication of the Owner.

The TOE also imports the requestor's public keyring along with incoming requests.

Handling permissions

The owner and the RoomAdministrator can edit the room membership permissions for each Room member, or invite new members.

The permissions of a Room that the TOE Drive is attached to are stored in Room blocks. These Room blocks are stored in both the TOE and the Clarabot Server in the form of a linked directed acyclic graph (so-called room config block chain). Each Room block is linked to the previous block by hashing and is signed by the Owner.

Handling incoming request

The TOE can only receive requests through the Clarabot Server. (The Clarabot Server used for relaying the requests can't read or manipulate them.)

In cases where the requestor is a Peer or the Owner (who sent the request from the webapp through the Clarabot Server), the context of the incoming request contains the Room id and the requesting account id. Based on them and with the help of different keys stored on the TOE, the request's payload can be decrypted. The applied cryptography (ECDH key exchange and AES-SIV encryption) also guarantees the authenticity of the requestor in case of successful decryption. After that, in accordance with the permissions of the requestor, the request will be accepted or rejected.

If the requestor is the Owner, and the request is an admin command to be executed, it will be handled based on the security configuration "remote_admin_policy":

- allow: all admin commands will be accepted,
- restrict: admin commands will be accepted if the request contains admin password that matches the admin password set on the TOE,
- deny: all admin commands will be rejected.

If the request requires a change of the TOE' state, then it must contain a unique session token, that has not been used before (the TOE keeps track of the currently valid tokens).

If the requestor is an Anonymous (account id is not present in the context) the request's payload can be decrypted with the help of keys stored on the TOE, received in the context and retrieved from the Clarabot Server. The applied cryptography (ECDH and AES-SIV) also guarantees that only the requestor has the key needed to decrypt the response. In case of successful decoding the request will only be accepted if anonymous access is allowed in the Room and the request requires read-only access.

There are settings the Owner may set that can enhance security. These local policy settings can override authorization of Peer and Anonymous requests:

- deny anonymous: If set, the TOE will reject Anonymous requests regardless of what the permission settings are in the related context of a room.

Response generation

The TOE can only send response through the Clarabot Server. (The Clarabot Server is used for relaying the responses can’t read or manipulate them.)

The TOE will decide dynamically if the request is cacheable or not. In both cases the response’s payload will be encrypted, although in a slightly different way.

In the first case (if the request is not cacheable), the response’s payload is encrypted on a peer-to-peer basis between accounts in the same way the requestor used, (based on common secret established during key exchange). The applied cryptography guarantees that only the requestor can decrypt the response.

In the other case (if the request is cacheable), the response’s payload is encrypted differently. Responses that would benefit from being cached are encrypted by the TOE using deterministically derived keys. The secret values generated by the TOE that are used for deriving the content keys never leave the TOE.

The Clarabot Server caches the deterministically encrypted response chunks, allowing multiple users to access the same content without encumbering the network uplink of the TOE. The applied cryptography guarantees that

- the requestor can decrypt the response’s payload,
- the Clarabot Server can utilize caching of encrypted data chunks (for a future shorter request round-trip-time),
- despite the caching feature, it is still guaranteed that the Clarabot Server won’t be able to serve a future request without consulting the TOE.

Cryptographic support

The TOE uses cryptography to

- decrypt and validate the integrity and authenticity of the requests retrieved from the Clarabot Server,
- encrypt, guarantee the authenticity and protect the integrity of the responses sent through Clarabot Server,
- validate the trust for accounts,
- protect the stored application data.

In order to achieve the above, the TOE uses different cryptographic keys and algorithms. Table 1 describes these keys:

Name	Purpose	Parameters	Generation	Storage
Encryption key (EK)	Encryption key for securing the Master key	32 byte, AES-SIV, expanded with HKDF to 64 bytes	Derived from password and server stored salt used in login process	Memory Part of the Private Keyring that is created for and kept during a login procedure
Master key (MK)	AES key for protecting Owner’s data	32 byte, AES-SIV, expanded with	Imported from Clarabot Server, protected by EK	Stored as a part of the Account Keyring, encrypted

Name	Purpose	Parameters	Generation	Storage
		HKDF to 64 bytes		by SEK in local storage
ed25519 -secret-key, public-key (ESK, EPK)	Owner’s Ed25519 key for signature	Ed25519	Imported from Clarabot Server, protected by MK	Stored as a part of the Account Keyring, encrypted by SEK in local storage
curve25519 - secret-key, public-key (CSK,CPK)	Owner’s Curve25519 key for ECDH key agreement	Curve25519	Imported from Clarabot Server protected by MK	Stored as a part of the Account Keyring, encrypted by SEK in local storage
Authentication key (AK)	Authentication key for login process	16 byte	Derived from password and server stored salt used in login process	Memory Part of the Private Keyring that is created for and kept during a login procedure
Session Encryption Key-source Part 1, 2, 3 (SEK1, SEK2, SEK3)	source of the Session Encryption Key (SEK)	SEK1: 16 bytes HKDF salt, SEK2: 16 bytes HKDF IKM SEK3: 32 bytes HKDF IKM	generated by TOE	SEK1 is stored in the TOE, SEK2 is stored on Clarabot Server, SEK3 is stored in the TOE
Session Encryption Key (SEK)	SEK is used to encrypt the secret keys in the Owner’s account keyring stored in the local storage (MK, ESK, CSK)	64 bytes, AES-SIV	Derived from SEK1, SEK2 and SEK3 using HKDF	Memory
Derived Master Key (DMK)	Derived master key for encrypting Owner’s request and for data encryption	64 bytes, AES-SIV	Derived from MK using HKDF	Memory
Derived P2P Key (DPK)	Key created by ECDH and HKDF for P2P encryption between the peer and the Owner	64 bytes, AES-SIV	Derived by using ECDH on Owner’s CSK and Peer’s CPK, and then deriving the result using HKDF (salt uses RK)	Memory
Anonymous Ephemeral Key pair (ASK, APK)	Ephemeral Curve25519 key, APK is included in the request for anonymous requests	Curve25519	(ASK, APK) is generated by Peer’s webapp, APK is imported in the Anonymous’s request	APK: Memory ASK: -
Room Key (RK)	Shared room key among all members. Used as salt in HKDF	32 bytes, HKDF salt	Imported from Clarabot server in encrypted form	Memory

Name	Purpose	Parameters	Generation	Storage
	when deriving P2P key (DPK)			
Local secrets (LK)	Local secrets used in HKDF for deriving keys on TOE (5 secret)	16 bytes, 32 bytes	Generated on the TOE	Local storage (config)
Part key (PK)	PK is used to encrypt the response chunks with AES-SIV. Sent in response setup part in encrypted form.	64 bytes, AES-SIV	Key derived from local secrets of the TOE and cache or transfer key as salt using HKDF	Memory

Table 1: Cryptographic keys used by TOE

Logging

The TOE generates a log of performed file-sharing and cryptographic operations (request import, response export, encryption, decryption, signing, signature verification, etc.). It records the time when the operation was performed and related account id. It also records failures (encryption failure, decryption failure, integrity verification failure, certificate validation failure, etc.). The operational environment provides a reliable time source and a text viewer application for logging function.

When desired, anonymous read-only access is allowed. In such cases, the log does not contain account id since Anonymous Nano users don't have one.

2. Conformance claims

2.1. CC conformance claim

This Security Target claims to be
“ISO/IEC 15408-2 conformant” and “ISO/IEC 15408-3 conformant”
and written according to the [CC1], [[CC2], [CC3], [CC4] and [CC5].

2.2. PP claim

This Security Target does not claim conformance to any Protection Profile.

2.3. Package claim

This ST conforms to assurance package EAL2 defined in [CC5] (package-name conformance).

2.4. Conformance rationale

As the ST does not claim conformance to a Protection Profile, a conformance rationale is not required.

3. Security Problem Definition

3.1. General

[CC1] and [CC3] defines assets as entities that the Owner of the TOE presumably places value upon. The term “asset” is used to describe the threats in the TOE operational environment.

3.1.1. Assets

The assets that need to be protected by the TOE are various forms of data.

R.OwnerKeyring: Keyring of the owner of the Nano Client

- master-key (MK) encrypted by EK (Encryption Key)
- ed25519-secret-key (ESK) encrypted by MK
- curve25519-secret-key (CSK) encrypted by MK

The confidentiality and integrity of the Owner’s keyring must be protected.

R.Roomblockchain: The membership and permissions in a room govern who can access what on resources of the computers by the attached Drives. The permissions are stored in room blocks, in a cryptographically linked directed acyclic graph, commonly referred to as a blockchain.
The integrity of the room’s blockchain must be protected.
It is also necessary to ensure the nonrepudiation of the blockchain.

R.PublicKeyring: The public keyring of a Peer
The integrity of the public keyrings must be protected.

R.NanoRequest: Nano request to be handled by the TOE.

- R.NanoRequest = (context, payload)
- context = (id of the requesting account, id of the room targeted by the request, type of request, various caching and data transfer control information)
- payload = requested detailed operation

The confidentiality and integrity of the payloads must be protected.
Nonrepudiation of payload origin shall be enforced.
The integrity of the context must be protected.

R.NanoResponse: Nano response handled by the TOE for a specific request.
The confidentiality and integrity of the responses must be protected.
Nonrepudiation of responses shall be enforced.

R.AuditLog: audit events recorded by the TOE.
The integrity of the audit events must be protected.

3.1.2. Subjects

S.Owner: the owner of the computer who installed the TOE. The owner also has a registered account in the Clarabot Server.

S.ClarabotServer: the Nano Clients and Nano web applications do not communicate on a peer-to-peer basis, but rather through centralized Clarabot Servers. The centralized servers provide fast and secure communication channels for all user commands and data transfer operations.

S.Peer: a user with a registered account in Clarabot Server.
Owners can invite one or more Peers to their room, giving them different role:

- membership (the contents of the drive will be available to them)
- administrator (see S.RoomAdministrator)

S.Anonymous: a user without a registered account in Clarabot Server, who can communicate with the owner's Nano Client if anonymous access is allowed.

S.RoomAdministrator: a user with a registered account in Clarabot Server, who is authorized to create new room blocks.

3.1.3. Threat agents of the TOE

Threat agents are:

A1 Attackers who have access to any communication channel over which the confidentiality and/or the integrity protected data (among them payloads of the requests and responses) are transferred, but have no valid account in the Nano system,

A2 Attackers who have or obtain valid account, but they ask for access beyond their privileges,

A3 Attackers (Anonymous) who have no valid account, and ask for access beyond their privileges,

A4 Attackers who have taken over the Clarabot Server and accessed it's database and messages going through it.

3.2. Threats

The following threats are defined for the TOE.

T.DATA_DISCLOSURE – loss of data confidentiality
An A1 attacker of one of the communication paths over which the data to be protected are transferred succeeds in accessing the content of the data, i.e. the attacker violates the confidentiality of the information included in the payloads.
The attack can for example be achieved by eavesdropping, recording encrypted data during the transfer and decoding the encrypted data.

T.DATA_MOD – loss of data integrity
An A1 attacker of one of the communication paths over which the data to be protected are transferred tampers with the payloads, i.e. replacing or modifying the content of the payloads in a way that is not detected.
The attack can for example be achieved by interrupting the transfer, modifying the content of the payload (including replacing the whole payload) and then re-constructing the integrity protection. Afterwards the modified payload is sent to the intended destination.

T.KEY – Key disclosure or modification
An A4 attacker succeeds in accessing or modifying secret keys when the keys are stored outside the TOE (included Clarabot Server) or in distribution.
The attack can for example be achieved by eavesdropping or modifying the content of key files during network transfer, or by accessing and modifying the key files when they are stored outside the TOE (e.g. on the Clarabot Server).

T.UNAUTHORIZED_ACCESS – Forged or illicit data retrieval/publishing requests
An A2 or A3 attacker forges requests that shall not be served by the TOE. If the TOE fails to adhere to the policies set up by the Owner the attacker may gain access to restricted content or may upload or change already existing content hosted by the TOE.

T.ROOM_BLOCKCHAIN_MOD – Forged or illicit permissions
An A1 or A4 attacker forges access rights stored in the Room blockchain.

3.3.Organizational Security Policies

The TOE and/or its operational environment shall comply with the following organizational security policies (OSPs) as security rules, procedures, practices or guidelines imposed by an organization upon its operation.

P.NONREPUDIATION
Non-repudiation of payload origin shall be enforced for Room blockchain, and all NanoRequests and NanoResponse.

P.ALGORITHM
Only cryptographic algorithms and key lengths determined by a standard, normative document should be used.

P.CRYPTO
The correct operation of crypto modules shall be verified.

P.LOGGING
Data sharing operations performed by the TOE shall be logged.

P.KEYSTORE
The key store shall be confidentiality and integrity protected.

3.4.Assumptions

This section specifies the assumptions that must be satisfied by the TOE operational environment.

A.PHYSICAL
The TOE is operated in a physically secure environment, i.e. no unauthorized persons have physical access to the TOE and its underlying system.

A.PLATFORM

The underlying operating system and hardware platform on which the TOE is installed work correctly (including that there are effective antivirus protection measures, and operating system access control is applied in order to restrict the accessibility e.g. the audit records or configuration files). The correct operation of their cryptographic functions are verified.

A.TIME
The TOE is provided with a reliable time source by its execution platform.

A.CSPRNG
A Cryptographically Secure Pseudorandom Number Generator provides random numbers for the TOE with sufficient amount of entropy.

A.AUDITVIEW
A text viewer application is available on the TOE to display the audit log.

A.KEYRING
Webapp owner performs secure keyring generation (during the registration).

A.ROOMADMINISTRATOR
It is assumed that the Room Administrators are trusted, competent and possesses the skills required for their tasks and are trained to conduct the activities they are responsible for.

A.USER
It is assumed that users (Owners and Peers) are trusted, they deliberately do not want to cause damage to the Nano system.
/A legitimate user that has been granted "write" access to a Nano Client could abuse their privilege to publish illicit content or malicious software. The simplest and most directly destructive action they could take is mass-deleting content from the Nano's storage. In a more sophisticated case they may use this to attack the Nano machine or other users that access the particular Nano./

A.PASSWORD
The password chosen by the Owner for protecting the keyring is of good quality and it is kept secret.

4. Security Objectives

This section identifies and defines the security objectives for the TOE and its environment. Security objectives reflect the stated intent and counter the identified threats, as well as comply with the identified organizational security policies and assumptions.

4.1.Security Objectives for the TOE

The following security objectives describe the security functions to be provided by the TOE.

O.CONFIDENTIALITY
The TOE shall provide mechanisms that protect the information of a transmitted data file such that its content is confidentiality-protected and only accessible for authorized users.
The confidentiality of cryptographic keys stored on the TOE should also be protected.

O.INTEGRITY
The TOE shall provide mechanisms that detect if an attacker has tampered with a transmitted data file (i.e. replacing or modifying the content of the data file).
The integrity of cryptographic keys stored on the TOE should also be protected.

O.NONREPUDIATION
The TOE shall provide a capability to generate evidence that can be used as a guarantee of the validity of the Room's blockchain, requests and responses (NanoRequests, NanoResponse).

O.ACCESSCONTROL
The TOE shall ensure that TOE resources are shared only on the basis of owner (or Room Administrator)-defined access permissions.

O.ACCOUNTABILITY
The TOE shall generate audit records for security-relevant events, recording the event details and the subject associated with the event.

O.ALGORITHM
The TOE shall only allow the use of cryptographic algorithms and key lengths recommended by a standard, authoritative document should be used.

4.2.Security Objectives for the Operational Environment

The following security objectives relate to the TOE environment. This includes client applications as well as the procedure for the secure operation of the TOE.

OE.PHYSICAL
The TOE is operated in a physically secure environment, i.e. no unauthorized persons have physical access to the TOE and its underlying system.

OE.PLATFORM
The underlying operating system and hardware platform on which the TOE is installed work correctly (including that there are effective antivirus protection measures, and operating system access control is applied in order to restrict the accessibility e.g. the audit records or configuration files). The correct operation of their cryptographic functions are verified.

OE.TIME
The TOE is provided with a reliable time source by its execution platform.

OE.CSPRNG
A Cryptographically Secure Pseudorandom Number Generator provides random numbers for the TOE with sufficient amount of entropy.

OE.AUDITVIEW
A text viewer application is available on the TOE to display the audit log.

OE.KEYRING
Clarabot web application performs secure keyring generation (during the registration) and sensitive key materials are stored encrypted on the server.

OE.TRUSTED_USERS
The Room Administrators are trusted, competent and possesses the skills required for their tasks and are trained to conduct the activities they are responsible for.
The Owner and Peers are trusted, they deliberately do not want to cause damage to the TOE.
The password chosen by the Owner for protecting the Account Keyring is of good quality and it is kept secret.

4.3.Security Objectives Rationale

4.3.1.Security objectives coverage

The Table 2 provides a mapping of the security objectives for the TOE to threats and policies, showing that each objective counters or enforces at least one threat or policy, respectively.

Objective for the TOE	Threats / OSPs
O. CONFIDENTIALITY	**T.DATA_DISCLOSURE** **T.KEY** **P.KEYSTORE**
O.INTEGRITY	**T.DATA_MOD** **T.KEY** **T.ROOM_BLOCKCHAIN_MOD** **P.NONREPUDIATION** **P.KEYSTORE**
O.NONREPUDIATION	**T.ROOM_BLOCKCHAIN_MOD** **P.NONREPUDIATION**
O.ACCESSCONTROL	**T.UNAUTHORIZED_ACCESS**
O.ACCOUNTABILITY	**P.LOGGING**
O.ALGORITHM	**P.ALGORITHM**

Table 2: Mapping of TOE security objectives to the problem definition

The Table 3 provides a mapping of the security objectives for the operational environment to assumptions, threats and policies, showing that each objective holds, counters or enforces at least one assumption, threat or policy, respectively.

Objective for the operational environment	Assumptions / Threats /OSPs
OE.PHYSICAL	**P.KEYSTORE** **A.PHYSICAL**
OE.PLATFORM	**P.CRYPTO** **A.PLATFORM**
OE.TIME	**P.LOGGING** **A.TIME**
OE.CSPRNG	**A.CSPRNG**
OE.AUDITVIEW	**P.LOGGING** **A.AUDITVIEW**
OE.KEYRING	**T.KEY** **P.ALGORITHM** **A.KEYRING**
OE.TRUSTED_USERS	**A.ROOMADMINISTRATOR** **A.USER** **A.PASSWORD** **T.UNAUTHORIZED_ACCESS**

Table 3: Mapping of security objectives for the Operational Environment to the problem definition

4.3.2.Security objectives sufficiency

The following rationale provides justification that the security objectives are suitable to counter each individual threat and that each security objective tracing back to a threat actually contributes to the mitigation of that threat.

Threat	Rationale for the security objectives
T.DATA_DISCLOSURE	O.CONFIDENTIALITY requires the TOE to provide mechanisms to protect the information of a transmitted data file. This diminishes T.DATA_DISCLOSURE by reducing the likelihood of a launched attack being successful; greater expertise and greater resources are needed from the attacker to perform attacks based on cryptoanalysis.
T.DATA_MOD	O.INTEGRITY requires the TOE to provide mechanisms to detect integrity violations of the file's content while transferring it over any unprotected communication channel. This diminishes T.DATA_MOD by reducing the likelihood of a launched attack being successful; greater expertise and greater resources are needed from the attacker to perform attacks based on cryptoanalysis.
T.KEY	O.CONFIDENTIALITY requires the TOE to provide mechanisms to protect the cryptographic keys stored on the TOE. O.INTEGRITY requires the TOE to provide mechanisms to protect the integrity of cryptographic keys stored on the TOE. OE.KEYRING requires the Clarabot web application to perform secure keyring generation during the registration. This diminish T.KEY as well.
T.UNAUTHORIZED_ACCESS	O.ACCESSCONTROL requires the TOE to ensure that TOE resources are shared only on the basis of owner and Room Administrator defined access permissions. This diminishes UNAUTHORIZED_ACCESS by reducing the likelihood of unauthorized access. O.ACCESSCONTROL is also supported by OE.TRUSTED_USERS which ensures that owners and Room Administrators are trusted, competent and possesses the skills required for their tasks.
T.ROOM_BLOCKCHAIN_MOD	O.INTEGRITY requires the TOE to provide mechanisms to protect the integrity of transmitted data file (including the Room blockchain). O.NONREPUDIATION requires the TOE to provide proof of the Room blockchain origin. They prevent an attacker from undetected forging access rights stored in the Room blockchain.

Table 4: Rationale for the security objectives

The following rationale provides justification that the security objective of the TOE is suitable to address each individual OSP and that each security objective tracing back to an OSP actually contributes in addressing the OSP.

OSP	Rationale for the OSPs
P.NONREPUDIATION	This OSP is addressed by O.NONREPUDIATION which requires the TOE to generate evidence that can be used as a guarantee of the validity of the Room's blockchain Nano requests and responses. O.INTEGRITY requires the TOE to provide mechanisms to detect integrity violations.
P.ALGORITHM	This OSP is addressed by O.ALGORITHM which ensures that only recommended cryptographic algorithms and key lengths are used. OE.KEYRING requires the Clarabot web application to perform secure keyring generation.
P.CRYPTO	This OSP is addressed by OE.PLATFORM which ensures that the correct operation of the platform's cryptographic functions are verified.
P.LOGGING	This OSP is addressed by O.ACCOUNTABILITY which ensures that audit records are created for security-relevant events. This OSP is supported by OT.TIME which ensures that the TOE is able to indicate the exact time in log records. This OSP is also supported by OE.AUDITVIEW which ensures that a text viewer application is available on the TOE to display the audit log.
P.KEYSTORE	This OSP is addressed by O.CONFIDENTIALITY and O.INTEGRITY which ensure that the confidentiality and the integrity of cryptographic keys stored on the TOE are protected. This OSP is also supported by OE.PHYSICAL which ensures that TOE is operated in a physically secure environment.

Table 5: Rationale for the OSPs

The following rationale provides justification that the security objectives of the TOE environment are suitable to address each individual assumption and that each security objective tracing back to an assumption actually contributes in addressing the assumption.

Assumption	Rationale for assumptions
A.PHYSICAL	Addressed by OE.PHYSICAL, which is identical to the assumption.
A.PLATFORM	Addressed by OE.PLATFORM, which is identical to the assumption.
A.TIME	Addressed by OE.TIME, which is identical to the assumption.
A.CSPRNG	Addressed by OE.CSPRNG, which is identical to the assumption.
A.AUDITVIEW	Addressed by OE.AUDITVIEW, which is identical to the assumption.
A.KEYRING	Addressed by OE.KEYRING, which is identical to the assumption.
A.ROOMADMINISTRATOR	Addressed by OE.TRUSTED_USERS, which contains the assumption.
A.USER	Addressed by OE.TRUSTED_USERS, which contains the assumption.
A.PASSWORD	Addressed by OE.TRUSTED_USERS, which contains the assumption.

Table 6: Rationale for assumptions

5. Extended components definition

There are no extended components.

6. Security requirements

6.1.Security functional requirements

The TOE satisfies the SFRs delineated in the following table. The rest of this chapter contains a description of each component and any related dependencies.

FAU_GEN.1	Audit data generation
FAU_GEN.2	User identity association
FCS_CKM.1/LK	Cryptographic key generation
FCS_CKM.1/SEK	Cryptographic key generation
FCS_COP.1/hash	Cryptographic operation (hashing)
FCS_COP.1/ECDH	Cryptographic operation (P2P key exchange)
FCS_COP.1/AES	Cryptographic operation (encrypt/decrypt)
FCS_COP.1/ECDSA	Cryptographic operation (signature creation/validation)
FCS_CKM.5/PBKDF2	Cryptographic key derivation
FCS_CKM.5/HKDF	Cryptographic key derivation
FDP_ITC.1/AccountKeyringImport	Import of user data without security attributes
FDP_IFC.1/AccountKeyringImport	Subset information flow control
FDP_IFF.1/AccountKeyringImport	Simple security attributes
FDP_ITC.2/PeerPublicKeyringImport	Import of user data with security attributes
FDP_IFC.1/PeerPublicKeyringImport	Subset information flow control
FDP_IFF.1/PeerPublicKeyringImport	Simple security attributes
FDP_ITC.2/RoomBlockchainImport	Import of user data with security attributes
FDP_IFC.1/RoomBlockchainImport	Subset information flow control
FDP_IFF.1/RoomBlockchainImport	Simple security attributes
FDP_ITC.2/CreateRoomBlock	Import of user data with security attributes
FDP_IFC.1/CreateRoomBlock	Subset information flow control
FDP_IFF.1/CreateRoomBlock	Simple security attributes
FDP_DAU.2/CreateRoomBlock	Data Authentication with Identity of Guarantor
FDP_ITC.2/OwnerRequestImport	Import of user data with security attributes
FDP_IFC.1/OwnerRequestImport	Subset information flow control
FDP_IFF.1/OwnerRequestImport	Simple security attributes
FDP_ITC.2/PeerRequestImport	Import of user data with security attributes
FDP_IFC.1/PeerRequestImport	Subset information flow control
FDP_IFF.1/PeerRequestImport	Simple security attributes
FDP_ITC.2/AnonymousRequestImport	Import of user data with security attributes
FDP_IFC.1/AnonymousRequestImport	Subset information flow control
FDP_IFF.1/AnonymousRequestImport	Simple security attributes
FDP_IFC.1/EncryptResponse	Subset information flow control
FDP_IFF.1/EncryptResponse	Simple security attributes
FDP_DAU.2/EncryptResponse	Data Authentication with Identity of Guarantor
FDP_ACC.1/AdminRequest	Subset access control
FDP_ACF.1/AdminRequest	Security attribute based access control
FDP_ACC.1/MutatingRequest	Subset access control
FDP_ACF.1/MutatingRequest	Security attribute based access control
FDP_ACC.1/DriveFileOperation	Subset access control
FDP_ACF.1/ DriveFileOperation	Security attribute based access control
FIA_UID.1	Timing of identification
FIA_UAU.1	Timing of authentication

FMT_MSA.1	Management of security attributes
FMT_MSA.3	Static attribute initialization
FMT_SMF.1	Specification of Management Functions
FMT_SMR.1	Security roles
FPT_TDC.1	Inter-TSF basic TSF data consistency
FTP_ITC.1/P2P	Inter-TSF trusted channel
FTP_ITC.1/Server	Inter-TSF trusted channel

Table 7: TOE Security Functional Requirements

Common Criteria allows several operations to be performed on functional requirements; refinement, selection, assignment, and iteration are defined in Chapter 8.2 of [CC2]. Each of these operations is used in this ST.

The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. The refinement operations are underlined in this ST.

The **selection** operation is used to select one or more options provided by the CC in stating a requirement. The selection operations are *italicized* in this ST.

The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password. The assignment operations are **[bolded]** in this ST.

The **iteration** operation is used when a component is repeated with varying operations. Iteration is denoted by showing a slash “/”, and the iteration indicator after the component identifier.

6.1.1. Security audit data generation (FAU)

FAU_GEN.1 (Audit data generation)

Hierarchical to: No other components.
Dependencies: FPT_STM.1 Reliable time stamps

FAU_GEN.1.1
The TSF shall be able to generate audit data of the following auditable events:

a) Start-up and shutdown of the audit functions;
b) All auditable events for the [*not specified*][1] level of audit;
c) [Remote access to Drive;
d) Nano remote configuration queried;
e) Configuration changes;
f) Room management operations][2]

FAU_GEN.1.2
The TSF shall record within the audit data at least the following information:

a) Date and time of the auditable event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event;
b) For each auditable event type, based on the auditable event definitions of the functional components included in the ST[3] [**human readable descriptive string about the related event**][4].

[1] [selection, choose one of: minimum, basic, detailed, not specified]
[2] [assignment: other specifically defined auditable events]
[3] [refinement: PP, PP-Module, functional package or ST]
[4] [assignment: other audit relevant information]

FAU_GEN.2 (User identity association)

Hierarchical to: No other components.
Dependencies: FAU_GEN.1 Audit data generation
FIA_UID.1 Timing of identification

FAU_GEN.2.1
For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

6.1.2. Cryptographic support (FCS)

FCS_CKM.1/LK (Cryptographic key generation)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or FCS_CKM.5 Cryptographic key derivation, or FCS_COP.1 Cryptographic operation]
FCS_CKM.3 Cryptographic key access
[FCS_RBG.1 Random bit generation, or FCS_RNG.1 Generation of random numbers]
FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM.1.1/LK
The TSF shall generate Local secrets (LK) cryptographic keys in accordance with a specified cryptographic key generation algorithm [**CSPRNG**][5] and specified cryptographic key sizes [**128, 256 bits**][6] that meet the following: [**8.1.5.2.1 in [SP800-57]**].[7]

FCS_CKM.1/SEK (Cryptographic key generation)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or FCS_CKM.5 Cryptographic key derivation, or FCS_COP.1 Cryptographic operation]
FCS_CKM.3 Cryptographic key access
[FCS_RBG.1 Random bit generation, or FCS_RNG.1 Generation of random numbers]
FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM.1.1/SEK
The TSF shall generate SEK1, SEK2 and SEK3 Session Encryption Key (SEK) cryptographic keys in accordance with a specified cryptographic key generation algorithm [**CSPRNG**][8] and specified cryptographic key sizes [**128, 128, 256 bits**][9] that meet the following: [**8.1.5.2.1 in [SP800-57]**].[10]

FCS_CKM.5/PBKDF2 (Cryptographic key derivation)

Hierarchical to: No other components.
Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM.5.1/PBKDF2

[5] [assignment: cryptographic key generation algorithm]
[6] [assignment: cryptographic key sizes]
[7] [assignment: list of standards]
[8] [assignment: cryptographic key generation algorithm]
[9] [assignment: cryptographic key sizes]
[10] [assignment: list of standards]

The TSF shall derive cryptographic keys [**AK and EK**][11] from [**password entered by the Owner, and salt send by Clarabot Server**][12] in accordance with a specified key derivation algorithm [**PBKDF2**][13] and specified cryptographic key sizes [**length of password and 16 bytes salt**][14] that meet the following: [**PKCS#5**][15].

FCS_CKM.5/HKDF (Cryptographic key derivation)

Hierarchical to: No other components.

Dependencies: [FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM.5.1/HKDF

The TSF shall derive cryptographic keys [

- **DMK**
- **DPK**
- **SEK**
- **PK**][16]

from [

- **MK**
- **P2P shared secret and RK as salt**
- **SEK1, SEK2 and SEK3 as salt**
- **SRK and (CK or TK) as a salt**][17]

in accordance with a specified key derivation algorithm [**HKDF**][18] and specified cryptographic key sizes [**256 bits**][19] that meet the following: [**RFC 5869**][20].

FCS_COP.1/hash (Cryptographic operation)

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 or
FCS_CKM.5 Cryptographic key derivation]
FCS_CKM.3 Cryptographic key access

FCS_COP.1.1/hash

The TSF shall perform **[cryptographic hash function]**[21] in accordance with a specified cryptographic algorithm **[SHA-256, SHA-512, SHA3-256]**[22] and cryptographic key sizes **[none]**[23] that meet the following: **[[TS 119312], [FIPS 180-4] and [FIPS 202]]**[24].

[11] [assignment: key type]
[12] [assignment: input parameters]
[13] [assignment: key derivation algorithm]
[14] [assignment: list of key sizes]
[15] [assignment: list of standards]
[16] [assignment: key type]
[17] [assignment: input parameters]
[18] [assignment: key derivation algorithm]
[19] [assignment: list of key sizes]
[20] [assignment: list of standards]
[21] [assignment: list of cryptographic operations]
[22] [assignment: cryptographic algorithm]
[23] [assignment: cryptographic key sizes]
[24] [assignment: list of standards]

Application Note:
hash functions are used:

- to hash function in HKDF (SHA-256)
- to digest Room blocks (SHA-512)
- to create SRK (serialized request key) (SHA3-256)

FCS_COP.1/ECDH (Cryptographic operation)

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 or
FCS_CKM.5 Cryptographic key derivation]
FCS_CKM.3 Cryptographic key access

FCS_COP.1.1/ECDH
The TSF shall perform **[Elliptic-curve Diffie–Hellman (ECDH) key exchange]**[25] in accordance with a specified cryptographic algorithm **[ECDH over Curve25519 curve, using C(0e, 2s, FFC DH), and C(1e, 1s, FFC DH) scheme]**[26] and cryptographic key sizes **[256 bits]**[27] that meet the following: **[[SP800-56A] and [SP800-186]]**[28].

Application Note:
ECDH key exchange produces a derived shared secret

- from Peer's CPK and Owner's CSK (0 ephemeral and 2 static)
- from ClarabotServer's SECPK and Owner's CSK (1 ephemeral and 1 static)
- from Anonymous's APK and Owner's CSK (1 ephemeral and 1 static)

FCS_COP.1/AES (Cryptographic operation)

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 or
FCS_CKM.5 Cryptographic key derivation]
FCS_CKM.3 Cryptographic key access

FCS_COP.1.1/AES
The TSF shall perform **[secure messaging - encryption and decryption]**[29] in accordance with a specified cryptographic algorithm **[AES in SIV mode]**[30] and cryptographic key sizes **[512 bits]**[31] that meet the following: **[[FIPS 197], [RFC 5297] and [SP800-38A]]**[32].

Application Note:
AES algorithm is used to:

- during initial Owner's account keyring import:
  - decrypt MK (using expanded EK)

[25] [assignment: list of cryptographic operations]
[26] [assignment: cryptographic algorithm]
[27] [assignment: cryptographic key sizes]
[28] [assignment: list of standards]
[29] [assignment: list of cryptographic operations]
[30] [assignment: cryptographic algorithm]
[31] [assignment: cryptographic key sizes]
[32] [assignment: list of standards]

    - decrypt ESK and CSK (using expanded MK)
    - decrypt session-id (using derived key from the shared secret of SECPK and CSK)
    - encrypt MK, ESK and CSK (using SEK)
- during subsequent Owner's account keyring import:
    - decrypt MK, ESK and CSK (using SEK)
- during incoming request:
    - decrypt request's payload (using DMK or DPK)
- during creating response:
    - encrypt SRK (using a local secret that is only available on the TOE)
    - encrypt PK and RAH (using P2P) – in the setup part
    - encrypt chunk (using PK) – in 0..n response
    - RK using the new Peer's CPK (during create a delta key)

FCS_COP.1/ECDSA (Cryptographic operation)

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 or
FCS_CKM.5 Cryptographic key derivation]
FCS_CKM.3 Cryptographic key access

FCS_COP.1.1/ECDSA
The TSF shall perform **[digital signature creation and verification]**[33] in accordance with a specified cryptographic algorithm **[ECDSA over Ed25519 curve]**[34] and cryptographic key sizes: [256 bits][35] that meet the following: **[[FIPS 186-4] and [SP800-186]]**[36].

Application Note:
ECDSA algorithm is used to:

- sign a CPK (with ESK)
- sign the fingerprint of the response (with ESK)
- validate a signature (with a public part of an Ed25519 key)

6.1.3. User data protection (FDP)

After a successful identification and authentication, the TOE essentially works automatically, enforcing the following information flow control SFPs:

- Keyring import (the TOE imports the owner's Account Keyring)
- Peer public keyring Import (the TOE imports a peer's public keyring)
- Room blockchain import (the TOE imports and verifies the blockchain of the room's permissions)
- Create a new room block (the TOE imports a new block and attach it to the blockchain)
- Owner request import (the TOE receives, decodes and handles a request from the Owner)

[33] [assignment: list of cryptographic operations]
[34] [assignment: cryptographic algorithm]
[35] [assignment: cryptographic key sizes]
[36] [assignment: list of standards]

- Peer request import (the TOE receives, decodes and handles a request from a Peer)
- Anonymous request import (the TOE receives, decodes and handles a request from an Anonymous)
- Encrypt response (the TOE sends an encrypted response to a request)

6.1.3.1 Owner's Account Keyring import

FDP_ITC.1/AccountKeyringImport (Import of user data without security attributes)

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
FMT_MSA.3 Static attribute initialization

FDP_ITC.1.1/AccountKeyringImport
The TSF shall enforce the [**Keyring import SFP**][37] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.1.2/AccountKeyringImport
The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

FDP_ITC.1.3/AccountKeyringImport
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [**none**][38].

FDP_IFC.1/AccountKeyringImport (Subset information flow control)

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/AccountKeyringImport
The TSF shall enforce [**Keyring import SFP**][39] on [**see Table 8**][40].

Account Keyring import information flow control SFP

Operations:

subject	information	operations
Owner	encrypted account keyring	import

Security attributes:

information	security attributes
encrypted keyring	AES-SIV authenticity check result

Rules:

Operations	security attributes	rules
import	AES-SIV authenticity check result	AES-SIV authenticity check result must be valid

Table 8: Account Keyring import SFP

[37] [assignment: access control SFP(s) and/or information flow control SFP(s)]
[38] [assignment: additional importation control rules]
[39] [assignment: information flow control SFP]
[40] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]

FDP_IFF.1/AccountKeyringImport (Simple security attributes)

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/AccountKeyringImport
The TSF shall enforce [**Keyring import SFP**][41] based on the following types of subject and information security attributes: [**see Table 9**][42].

FDP_IFF.1.2/AccountKeyringImport
The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**see Table 9**][43].

FDP_IFF.1.3/AccountKeyringImport
The TSF shall enforce the [**none**][44].

FDP_IFF.1.4/AccountKeyringImport
The TSF shall explicitly authorise an information flow based on the following rules: [**AES-SIV authenticity check result is valid**][45].

FDP_IFF.1.5/AccountKeyringImport
The TSF shall explicitly deny an information flow based on the following rules: [**AES-SIV authenticity check result is invalid**][46].

6.1.3.2 Peer's public keyring import

FDP_ITC.2/PeerPublicKeyringImport (Import of user data with security attributes)

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]
FPT_TDC.1 Inter-TSF basic TSF data consistency

FDP_ITC.2.1/PeerPublicKeyringImport
The TSF shall enforce the **[Peer public keyring import SFP**][47] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/PeerPublicKeyringImport
The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/PeerPublicKeyringImport
The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

[41] [assignment: information flow control SFP]
[42] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]
[43] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]
[44] [assignment: additional information flow control SFP rules]
[45] [assignment: rules, based on security attributes, that explicitly authorize information flows]
[46] [assignment: rules, based on security attributes, that explicitly deny information flows]
[47] [assignment: access control SFP(s) and/or information flow control SFP(s)]

FDP_ITC.2.4/PeerPublicKeyringImport
The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/PeerPublicKeyringImport
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **[none]**[48].

FDP_IFC.1/PeerPublicKeyringImport (Subset information flow control)

Hierarchical to: No other components.
Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/PeerPublicKeyringImport
The TSF shall enforce **[Peer public keyring import SFP]**[49] on [**see: Table 9**][50]

Peer public keyring import SFP

Operations:

subject	information	operations
Peer	public keyring /including: • account id, CPK, EPK, • signature of CPK signed by ESK, • trust fingerprint	import

Security attributes:

subject	security attributes
peer	account id
information	**security attributes**
EPK (trust root)	trust fingerprint
CPK	signature of CPK signed by ESK

Rules:

Operations	security attributes	rules
import	trust fingerprint	usage of trust fingerprint is mandatory in the evaluated version: the calculated value must match the imported value
import	signature of CPK signed by ESK	the signature validity check must be valid

Table 9: Peer public Keyring import SFP

FDP_IFF.1/PeerPublicKeyringImport (Simple security attributes)

Hierarchical to: No other components.
Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/PeerPublicKeyringImport
The TSF shall enforce [**Peer public keyring import SFP]**[51] based on the following types of subject and information security attributes: [**see Table 9**][52].

FDP_IFF.1.2/PeerPublicKeyringImport
The TSF shall permit an information flow between a controlled subject and controlled information

[48] additional importation control rules
[49] [assignment: information flow control SFP]
[50] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]
[51] [assignment: information flow control SFP]
[52] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]

via a controlled operation if the following rules hold: **[see Table 9]**[53]**.**

FDP_IFF.1.3/PeerPublicKeyringImport
The TSF shall enforce the [**none**][54].

FDP_IFF.1.4/PeerPublicKeyringImport
The TSF shall explicitly authorise an information flow based on the following rules: **[there is a TF trust fingerprint in the imported keyring and the re-calculated TF' value match the imported TF value)]**[55].

FDP_IFF.1.5/PeerPublicKeyringImport
The TSF shall explicitly deny an information flow based on the following rules: **[**

- **the public keyring does not contain a trust fingerprint**][56].

6.1.3.3 Room blockchain import

FDP_ITC.2/RoomBlockchainImport (Import of user data with security attributes)

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]
FPT_TDC.1 Inter-TSF basic TSF data consistency

FDP_ITC.2.1/RoomBlockchainImport
The TSF shall enforce the **[Room blockchain import SFP]**[57] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/RoomBlockchainImport
The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/RoomBlockchainImport
The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/RoomBlockchainImport
The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/RoomBlockchainImport
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **[none]**[58].

FDP_IFC.1/RoomBlockchainImport (Subset information flow control)

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

[53] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]

[54] [assignment: additional information flow control SFP rules]

[55] [assignment: rules, based on security attributes, that explicitly authorize information flows]

[56] [assignment: rules, based on security attributes, that explicitly deny information flows]

[57] [assignment: access control SFP(s) and/or information flow control SFP(s)]

[58] additional importation control rules

FDP_IFC.1.1/RoomBlockchainImport
The TSF shall enforce **[Room blockchain import SFP]**[59] on [**see Table 10**][60].

Room blockchain import SFP

Operations:

subject	**information**	**operations**
Owner	room blockchain	import

Security attributes:

subject	**security attributes**
Owner	EPK (stored in the public keyring)
information	**security attributes**
i^{th} block in the blockchain (i=1,…n)	signature of the i^{th} block by Owner's ESK (contained by the block)
i^{th} block in the blockchain (i=2,…n)	the SHA2-512 hash value of the $(i-1)^{th}$ block (contained by the block)
the blockchain	the SHA2-512 hash value of the n^{th} block (stored in the TOE configuration)

Rules:

Operations	**security attributes**	**rules**
import	signature of the i^{th} block by Owner's ESK	for all i (i=1,…n): the signature verification result must be valid
import	the SHA2-512 hash value of the $(i-1)^{th}$ block	for all i (i=2,…n): the calculated hash value must match the imported hash value
import	the SHA2-512 hash value of the n^{th} block	the hash value must match the stored value

Table 10: Room blockchain import SFP

FDP_IFF.1/RoomBlockchainImport (Simple security attributes)

Hierarchical to: No other components.
Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/RoomBlockchainImport
The TSF shall enforce **[Room blockchain import information flow control SFP]**[61] based on the following types of subject and information security attributes: [**see Table 10**][62].

FDP_IFF.1.2/RoomBlockchainImport
The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**see Table 10**][63].

FDP_IFF.1.3/RoomBlockchainImport
The TSF shall enforce the [**none**][64].

FDP_IFF.1.4/RoomBlockchainImport

[59] [assignment: information flow control SFP]
[60] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]
[61] [assignment: information flow control SFP]
[62] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]
[63] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]
[64] [assignment: additional information flow control SFP rules]

The TSF shall explicitly authorise an information flow based on the following rules: [**none**][65].

FDP_IFF.1.5/RoomBlockchainImport
The TSF shall explicitly deny an information flow based on the following rules: [**none]**[66].

6.1.3.4 Create a new room block

FDP_ITC.2/CreateRoomBlock (Import of user data with security attributes)

Hierarchical to: No other components.
Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]
FPT_TDC.1 Inter-TSF basic TSF data consistency

FDP_ITC.2.1/CreateRoomBlock
The TSF shall enforce the **[Create room block SFP**][67] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/CreateRoomBlock
The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/CreateRoomBlock
The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/CreateRoomBlock
The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/CreateRoomBlock
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **[none]**[68].

FDP_IFC.1/CreateRoomBlock (Subset information flow control)

Hierarchical to: No other components.
Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/CreateRoomBlock
The TSF shall enforce **[Create room block SFP]**[69] on [**see Table 11**][70].

[65] [assignment: rules, based on security attributes, that explicitly authorize information flows]
[66] [assignment: rules, based on security attributes, that explicitly deny information flows]
[67] [assignment: access control SFP(s) and/or information flow control SFP(s)]
[68] additional importation control rules
[69] [assignment: information flow control SFP]
[70] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]

Create room block SFP

Operations:

subject	information	operations
RoomAdministrator	new room block request	create

Security attributes:

subject	security attributes
RoomAdministrator	account id membership flags linked to account id
information	**security attributes**
new room block's payload	-

Rules:

Operations	security attributes	rules
create	account id	the account id must be present in the TOE local store
create	membership flags linked to account id	the membership flags must be: "active", "administrator"

Table 11: Create room block SFP

FDP_IFF.1/CreateRoomBlock (Simple security attributes)

Hierarchical to: No other components.
Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/CreateRoomBlock
The TSF shall enforce **[Create room block SFP**][71] based on the following types of subject and information security attributes: [**see Table 11**][72].

FDP_IFF.1.2/CreateRoomBlock
The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**see Table 11**][73].

FDP_IFF.1.3/CreateRoomBlock
The TSF shall enforce the [**none**][74].

FDP_IFF.1.4/CreateRoomBlock
The TSF shall explicitly authorise an information flow based on the following rules: [**none**][75].

FDP_IFF.1.5/CreateRoomBlock
The TSF shall explicitly deny an information flow based on the following rules: [**none]**[76].

[71] [assignment: information flow control SFP]

[72] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]

[73] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]

[74] [assignment: additional information flow control SFP rules]

[75] [assignment: rules, based on security attributes, that explicitly authorize information flows]

[76] [assignment: rules, based on security attributes, that explicitly deny information flows]

FDP_DAU.2/CreateRoomBlock (Data Authentication with Identity of Guarantor)

Hierarchical to: FDP_DAU.1 Basic Data Authentication
Dependencies: FIA_UID.1 Timing of identification

FDP_DAU.2.1/CreateRoomBlock
The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of **[the created new room block and the modified room blockchain]**[77]**.**

FDP_DAU.2.2/CreateRoomBlock
The TSF shall provide **[Owner and Clarabot Server]**[78] with the ability to verify evidence of the validity of the indicated information and the identity of the user that generated the evidence.

6.1.3.5 Owner Request import

FDP_ITC.2/OwnerRequestImport (Import of user data with security attributes)

Hierarchical to: No other components.
Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]
FPT_TDC.1 Inter-TSF basic TSF data consistency

FDP_ITC.2.1/OwnerRequestImport
The TSF shall enforce the [**Owner request import SFP**][79] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/OwnerRequestImport
The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/OwnerRequestImport
The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/OwnerRequestImport
The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/OwnerRequestImport
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [**none**][80].

FDP_IFC.1/OwnerRequestImport (Subset information flow control)

Hierarchical to: No other components.
Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/OwnerRequestImport
The TSF shall enforce [**Owner request import SFP**]**[81]** on [

a) subject: Owner;
b) information: request;

[77] [assignment: list of objects or information types]
[78] [assignment: list of subjects]
[79] [assignment: access control SFP(s) and/or information flow control SFP(s)]
[80] additional importation control rules
[81] [assignment: information flow control SFP]

c) operations:

- **decrypt the request (according to Nano Own Cryptographic Context),**
- **pass it over for processing the decrypted request**][82]

FDP_IFF.1/OwnerRequestImport (Simple security attributes)

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/OwnerRequestImport
The TSF shall enforce **[Owner request import SFP]**[83] based on the following types of subject and information security attributes: **[**

a) subject security attributes:

- **account id (from the context)**
- **Account Keyring (stored by the TOE);**

b) information security attributes:

- **Room id (from the context)]**[84].

FDP_IFF.1.2/OwnerRequestImport
The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**after decrypting the request according to Nano Own Cryptographic Context (based on security attributes) the AES-SIV authenticity check result is valid**][85].

FDP_IFF.1.3/OwnerRequestImport
The TSF shall enforce the [**none**][86].

FDP_IFF.1.4/OwnerRequestImport
The TSF shall explicitly authorise an information flow based on the following rules: **[none**][87].

FDP_IFF.1.5/OwnerRequestImport
The TSF shall explicitly deny an information flow based on the following rules: **[there is not account id, or account id is not the Owner**][88].

6.1.3.6 Peer Request import

FDP_ITC.2/PeerRequestImport (Import of user data with security attributes)

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]
FPT_TDC.1 Inter-TSF basic TSF data consistency

[82] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]
[83] [assignment: information flow control SFP]
[84] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]
[85] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]
[86] [assignment: additional information flow control SFP rules]
[87] [assignment: rules, based on security attributes, that explicitly authorize information flows]
[88] [assignment: rules, based on security attributes, that explicitly deny information flows]

FDP_ITC.2.1/PeerRequestImport
The TSF shall enforce the [**Peer request import SFP**][89] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/PeerRequestImport
The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/PeerRequestImport
The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/PeerRequestImport
The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/PeerRequestImport
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [**none**][90].

FDP_IFC.1/PeerRequestImport (Subset information flow control)

Hierarchical to: No other components.
Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/PeerRequestImport
The TSF shall enforce [**Peer request import SFP**][91] on [

a) subject: Peer;

b) information: request;

c) operations:

- **decrypt the request (according to Nano Cryptographic Context),**
- **pass it over for processing the decrypted request**][92]

FDP_IFF.1/PeerRequestImport (Simple security attributes)

Hierarchical to: No other components.
Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/PeerRequestImport
The TSF shall enforce **[Peer request import SFP]**[93] based on the following types of subject and information security attributes: **[**

a) subject security attributes:

- **account id (from the context)**
- **public keyring related to the account id (stored by the TOE);**

b) information security attributes:

- **Room id (from the context)]**[94].

[89] [assignment: access control SFP(s) and/or information flow control SFP(s)]

[90] additional importation control rules

[91] [assignment: information flow control SFP]

[92] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]

[93] [assignment: information flow control SFP]

[94] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]

FDP_IFF.1.2/PeerRequestImport
The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**after decrypting the request according to Nano Cryptographic Context (based on security attributes) the AES-SIV authenticity check result is valid**][95].

FDP_IFF.1.3/PeerRequestImport
The TSF shall enforce the [**Peer public keyring import SFP**][96] if the account id related public keyring does not stored by the TOE yet.

FDP_IFF.1.4/PeerRequestImport
The TSF shall explicitly authorise an information flow based on the following rules: **[none]**[97].

FDP_IFF.1.5/PeerRequestImport
The TSF shall explicitly deny an information flow based on the following rules: **[there is not account id, or account id is the Owner**][98].

6.1.3.7 AnonymousRequest import

FDP_ITC.2/AnonymousRequestImport (Import of user data with security attributes)
Hierarchical to: No other components.
Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
[FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]
FPT_TDC.1 Inter-TSF basic TSF data consistency

FDP_ITC.2.1/AnonymousRequestImport
The TSF shall enforce the [**Anonymous request import SFP**][99] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/AnonymousRequestImport
The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/AnonymousRequestImport
The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/AnonymousRequestImport
The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/AnonymousRequestImport
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [**none**][100].

[95] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]
[96] [assignment: additional information flow control SFP rules]
[97] [assignment: rules, based on security attributes, that explicitly authorize information flows]
[98] [assignment: rules, based on security attributes, that explicitly deny information flows]
[99] [assignment: access control SFP(s) and/or information flow control SFP(s)]
[100] additional importation control rules

FDP_IFC.1/AnonymousRequestImport (Subset information flow control)

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/AnonymousRequestImport

The TSF shall enforce [**Anonymous request import SFP]**[101] on [

a) subject: Anonymous;

b) information: request;

c) operations:

- **decrypt the request (according to Nano Cryptographic Context),**
- **pass it over for processing the decrypted request**][102]

FDP_IFF.1/AnonymousRequestImport (Simple security attributes)

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/AnonymousRequestImport

The TSF shall enforce **[Anonymous request import SFP]**[103] based on the following types of subject and information security attributes: **[**

a) subject security attributes:

- **APK (from the context);**

b) information security attributes:

- **Room id (from the context)]**[104]**.**

FDP_IFF.1.2/AnonymousRequestImport

The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [**after decrypting the request according to Nano Cryptographic Context (based on security attributes) the AES-SIV authenticity check result is valid**][105].

FDP_IFF.1.3/AnonymousRequestImport

The TSF shall enforce the [**none**][106].

FDP_IFF.1.4/AnonymousRequestImport

The TSF shall explicitly authorise an information flow based on the following rules: **[none**][107].

FDP_IFF.1.5/AnonymousRequestImport

The TSF shall explicitly deny an information flow based on the following rules: **[there is an account id in the context, or there is no APK in the context**][108].

[101] [assignment: information flow control SFP]

[102] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]

[103] [assignment: information flow control SFP]

[104] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]

[105] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]

[106] [assignment: additional information flow control SFP rules]

[107] [assignment: rules, based on security attributes, that explicitly authorize information flows]

[108] [assignment: rules, based on security attributes, that explicitly deny information flows]

6.1.3.8 Encrypt response

FDP_IFC.1/EncryptResponse (Subset information flow control)

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/EncryptResponse

The TSF shall enforce [**Encypt response SFP**][109] on
[**see Table 8**][110].

Encrypt response control SFP

Operations:

subject	**information**	**operations**
Owner	encrypted response	encrypting and sending a response

Security attributes:

subject	**security attributes**
Owner	account id
information	**security attributes**
encrypted response	cache flag (response is cacheable / not cacheable)

Rules:

Operations	**security attributes**	**rules**
send (to Clarabot Server)	cache flag	If the response is cacheable: CK is used for PK generation. If the response is non-cacheable: TK is used for PK generation.

Table 12: Encrypt response SFP

FDP_IFF.1/EncryptResponse (Simple security attributes)

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control
FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/EncryptResponse

The TSF shall enforce [**Encrypt response information flow control SFP**][111] based on the following types of subject and information security attributes: [**see Table 12**][112].

FDP_IFF.1.2/EncryptResponse

The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **[see Table 12]**[113].

FDP_IFF.1.3/EncryptResponse

The TSF shall enforce the **[none]**[114].

FDP_IFF.1.4/EncryptResponse

[109] [assignment: information flow control SFP]

[110] [assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP]

[111] [assignment: information flow control SFP]

[112] [assignment: list of subjects and information controlled under the indicated SFP, and for each, the security attributes]

[113] [assignment: for each operation, the security attribute-based relationship that hold between subject and information security attributes]

[114] [assignment: additional information flow control SFP rules]

The TSF shall explicitly authorise an information flow based on the following rules: [**none**][115].

FDP_IFF.1.5/EncryptResponse
The TSF shall explicitly deny an information flow based on the following rules: **[none]**[116].

FDP_DAU.2/EncryptResponse (Data Authentication with Identity of Guarantor)
Hierarchical to: FDP_DAU.1 Basic Data Authentication
Dependencies: FIA_UID.1 Timing of identification

FDP_DAU.2.1/EncryptResponse
The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of **[responses]**[117]**.**

FDP_DAU.2.2/EncryptResponse
The TSF shall provide **[requestor Peer]**[118] with the ability to verify evidence of the validity of the indicated information and the identity of the user that generated the evidence.

The TOE enforces the following access control SFPs:

- Admin request
  (enforce that admin requests can only be issued according to the security configuration)
- Mutating request
  (enforce that the request that changes the internal state of the TOE must contain an unique session token)
- Drive file operation
  (enforce that in the requests the selected file/path must be included in the Drive).

6.1.3.9 Admin request

FDP_ACC.1/AdminRequest (Subset access control)
Hierarchical to: No other components.
Dependencies: FDP_ACF.1 Security attribute based access control
FDP_ACC.1.1/AdminRequest
The TSF shall enforce the **[Admin request SFP]**[119] on **[see Table 13]**[120].

[115] [assignment: rules, based on security attributes, that explicitly authorize information flows]
[116] [assignment: rules, based on security attributes, that explicitly deny information flows]
[117] [assignment: list of objects or information types]
[118] [assignment: list of subjects]
[119] [assignment: access control SFP]
[120] [assignment: list of subjects, objects, and operations among subjects and objects covered by the SFP]

Admin request SFP

Operations:

subject	object	operations
Owner	admin command in the request	handling the admin command (accept/reject)

Security attributes:

subject	security attributes
Owner	account id, admin password (optional)
object	**security attributes**
admin command in the request	"remote_admin_policy" configuration parameter

Rules:

Operations	security attributes	rules
handling the admin command in the request: (pass for further examination / reject)	"remote_admin_policy" configuration parameter	If "remote_admin_policy" is • allow: the admin command will be passed for further examination, • deny: the admin command will be rejected, • restrict: admin command must contain the admin password that matches the admin password set on the TOE.
handling the admin command in the request: accept/reject	account id, password (optional)	If "remote_admin_policy" is restrict: • the admin password must be match the password set on the TOE.

Table 13: Admin request SFP

FDP_ACF.1/AdminRequest (Security attribute based access control)

Hierarchical to: No other components.

Dependencies: FDP_ACC.1 Subset access control
FMT_MSA.3 Static attribute initialization

FDP_ACF.1.1/AdminRequest
The TSF shall enforce the **[Admin request SFP]**[121] to objects based on the following: **[see Table 13]**[122].

FDP_ACF.1.2/AdminRequest
The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **[see Table 13]**[123].

FDP_ACF.1.3/AdminRequest
The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **[none]**[124].

[121] [assignment: access control SFP]
[122] [assignment: list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes]
[123] [assignment: rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects]
[124] [assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]

FDP_ACF.1.4/AdminRequest
The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[none]**[125].

6.1.3.10 Mutating request

FDP_ACC.1/MutatingRequest (Subset access control)

Hierarchical to: No other components.
Dependencies: FDP_ACF.1 Security attribute based access control

FDP_ACC.1.1/MutatingRequest
The TSF shall enforce the **[Mutating request SFP]**[126] on **[see Table 14]**[127].

Mutating request SFP

Operations:

subject	**object**	**operations**
Peer or Owner	mutating request (included a session token)	handling the mutating command (accept/reject)

Security attributes:

subject	**security attributes**
Peer or Owner	membership flags linked to account id
object	**security attributes**
mutating request	session token (included in the request) request type

Rules:

Operations	**security attributes**	**rules**
handling the mutating command (accept/reject)	membership flags	the membership flags must be: • "active" • in case of a file modification request: "Drive write" • in case of a new file upload request: "Drive upload" • in case of a new folder insert request: "Drive adhoc",
handling the mutating command (accept/reject)	session token	session token uniqueness check must be successful
handling the mutating command (accept/reject)	request type	if request type is a mutating request type, then the session token check is mandatory

Table 14: Mutating request SFP

FDP_ACF.1/MutatingRequest (Security attribute based access control)

Hierarchical to: No other components.
Dependencies: FDP_ACC.1 Subset access control
FMT_MSA.3 Static attribute initialization

[125][assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]
[126] [assignment: access control SFP]
[127] [assignment: list of subjects, objects, and operations among subjects and objects covered by the SFP]

FDP_ACF.1.1/MutatingRequest
The TSF shall enforce the **[Mutating request SFP]**[128] to objects based on the following: **[see: Table 14]**[129].

FDP_ACF.1.2/MutatingRequest
The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **[see: Table 14]**[130].

FDP_ACF.1.3/MutatingRequest
The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **[none]**[131].

FDP_ACF.1.4/MutatingRequest
The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[none]**[132].

6.1.3.11 Drive file operation

FDP_ACC.1/DriveFileOperation (Subset access control)

Hierarchical to: No other components.
Dependencies: FDP_ACF.1 Security attribute based access control

FDP_ACC.1.1/DriveFileOperation
The TSF shall enforce the **[Drive file operation SFP]**[133] on **[see Table 15]**[134].

Drive file operation access control SFP

Operations:

subject	object	operations
all	the file/folder affected by the request	requested operation in the request (read/write/….)

Security attributes:

subject	security attributes
all	-
object	**security attributes**
the file/folder affected by the request	requested Drive selected file/path in the request

Rules:

Operations	security attributes	rules
requested operation in the request	selected file/path in the request	the selected file/path must be included in the Drive.

Table 15: Drive file operation SFP

[128] [assignment: access control SFP]
[129] [assignment: list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes]
[130] [assignment: rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects]
[131] [assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]
[132] [assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]
[133] [assignment: access control SFP]
[134] [assignment: list of subjects, objects, and operations among subjects and objects covered by the SFP]

FDP_ACF.1/DriveFileOperation (Security attribute based access control)

Hierarchical to: No other components.
Dependencies: FDP_ACC.1 Subset access control
FMT_MSA.3 Static attribute initialization

FDP_ACF.1.1/DriveFileOperation
The TSF shall enforce the **[Drive file operation SFP]**[135] to objects based on the following: **[see: Table 15]**[136].

FDP_ACF.1.2/DriveFileOperation
The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **[see Table 15]**[137].

FDP_ACF.1.3/DriveFileOperation
The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **[none]**[138].

FDP_ACF.1.4/DriveFileOperation
The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[none]**[139].

6.1.4. Identification and authentication (FIA)

FIA_UID.1 (Timing of identification)

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

FIA_UID.1.1
The TSF shall allow: [**none**][140] on behalf of the user to be performed before the user is identified.

FIA_UID.1.2
The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

FIA_UAU.1 (Timing of authentication)

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

FIA_UAU.1.1
The TSF shall allow: [**none**][141] on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2
The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

[135] [assignment: access control SFP]
[136] [assignment: list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes]
[137] [assignment: rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects]
[138] [assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]
[139] [assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]
[140] [assignment: list of TSF-mediated actions]
[141] [assignment: list of TSF-mediated actions]

6.1.5. Security management (FMT)

FMT_MSA.1 (Management of security attributes)

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
FMT_SMR.1 Security roles
FMT_SMF.1 Specification of Management Functions

FMT_MSA.1.1

The TSF shall enforce the [**SFPs in Table 16**][142] to restrict the ability to [***see Table 16***][143] the security attributes [**see Table 16**][144] to [**Owner**s.

SFP	Security attribute	Operation
Account Keyring import	AES-SIV authenticity check result	none
Peer public keyring import	account id	none
	trust fingerprint	none
	SHA2-256 hash of the EPK	none
	signature of CPK signed by ESK	none
Room blockchain import	EPK (stored in the public keyring)	none
	signature of the i^{th} block by Owner's ESK (contained by the block)	none
	the SHA2-512 hash value of the $(i-1)^{th}$ block (contained by the block)	none
	the SHA2-512 hash value of the n^{th} block (stored in the TOE configuration)	none
Create room block	account id	none
	membership flags linked to account id	none
Owner Request Import	account id	none
	Account Keyring	none
	Room id	none
Peer Request Import	account id	none
	public keyring	none
	Room id	none
Anonymous Request Import	APK	none
	Room id	none
Encrypt Response SFP	account id	none
	cache flag	none
Admin request SFP	account id	none
	admin password (optional)	modify
	"remote_admin_policy" configuration parameter	none
Mutating Request SFP	membership flags	none
	session token	none
	request type	none
Drive file operation SFP	selected file/path in the request	none
	selected file/path in the request	none
	requested Drive	none

Table 16: Security attributes

[142] [assignment: access control SFP(s), information flow control SFP(s)]
[143] [selection: change_default, query, modify, delete, [assignment: other operations]]
[144] [assignment: list of security attributes]

FMT_MSA.3 (Static attribute initialization)

Hierarchical to: No other components.

Dependencies: FMT_MSA.1 Management of security attributes
FMT_SMR.1 Security roles

FMT_MSA.3.1

The TSF shall enforce the [**all SFP**][145] to provide [*restrictive*][146] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2

The TSF shall allow the [**nobody**][147] to specify alternative initial values to override the default values when an object or information is created.

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: [

- **Change (Owner) password**
- **Set admin password**
- **Change admin password**
- **Reset Nano Client**
- **Create Drive**
- **View audit records**
- **Create Room block (Configure content editing and content sharing privileges)**
- **Update deny anonymous**
- **Remote drive management**][148].

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 **[Owner, Peer, Anonymous, RoomAdministrator]**[149].

FMT_SMR.1.2

The TSF shall be able to associate users with roles.

[145] [assignment: access control SFP, information flow control SFP]

[146] [selection: choose one of: restrictive, permissive, [assignment: other property]]

[147] [assignment: the authorized identified roles]

[148] [assignment: list of management functions to be provided by the TSF]

[149] [assignment: the authorized identified roles]

6.1.6. Protection of the TSF (FPT)

FPT_TDC.1 **(Inter-TSF basic TSF data consistency)**

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

FPT_TDC.1.1
The TSF shall provide the capability to consistently interpret **[**

- **account keyring,**
- **public keyring,**
- **room blockchain**
- **request's context and payload]**[150]

when shared between the TSF and Clarabot Server[151].

FPT_TDC.1.2
The TSF shall use **[integrity and authenticity on data]**[152] when interpreting the TSF data from Clarabot Server[153].

[150] [assignment: list of TSF data types]
[151] [refinement: another trusted IT product]
[152] [assignment: list of interpretation rules to be applied by the TSF]
[153] [refinement: another trusted IT product]

6.1.7. Trusted path/channels (FTP)

FTP_ITC.1/P2P (Inter-TSF trusted channel)

Hierarchical to: No other components.

Dependencies: No dependencies.

FTP_ITC.1.1/P2P

The TSF shall provide a communication channel between itself and Peer's Clarabot Client (through the Clarabot Server)[154] 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/P2P

The TSF shall permit [*Clarabot Server*][155] [156] to initiate communication via the trusted channel.

FTP_ITC.1.3/P2P

The TSF shall initiate communication via the trusted channel for [

1. **import Owner's request**
2. **import Peer's request,**
3. **import Anonymous request**][157].

FTP_ITC.1/Server (Inter-TSF trusted channel)

Hierarchical to: No other components.

Dependencies: No dependencies.

FTP_ITC.1.1/Server

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/Server

The TSF shall permit [*the TSF, Clarabot Server*][158] [159] to initiate communication via the trusted channel.

FTP_ITC.1.3/Server

The TSF shall initiate communication via the trusted channel for [

1. **import Account Keyring**
2. **import Peer's public keyring,**
3. **import room blockchain**
4. **import a CreateRoomBlock request**][160].

[154] [refinement: another trusted IT product]
[155] [selection: the TSF, another trusted IT product]
[156] [refinement: another trusted IT product]
[157] [assignment: list of functions for which a trusted channel is required]
[158] [selection: the TSF, another trusted IT product]
[159] [refinement: another trusted IT product]
[160] [assignment: list of functions for which a trusted channel is required]

6.2.Security assurance requirements

The evaluation assurance level for this ST is EAL2, the following table lists the assurance requirements.

Class Assurance	**Assurance components**
ADV: Development	ADV_ARC.1 Architectural Design with domain separation and nonbypassability
	ADV_FSP.2 Security-enforcing functional specification
	ADV_TDS.1 Basic design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.2 Use of the CM system
	ALC_CMS.2 Parts of the TOE CM coverage
	ALC_DEL.1 Delivery procedures
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	ASE_INT.1 ST introduction
	ASE_OBJ.2 Security objectives
	ASE_REQ.2 Derived security requirements
	ASE_SPD.1 Security problem definition
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_COV.1 Evidence of coverage
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing – sample
AVA: Vulnerability assessment	AVA_VAN.2 Vulnerability analysis

Table 17: TOE Security Assurance Requirements

6.3.Security requirements rationale

6.3.1. Security functional requirements coverage

Security Functional Requirement	O.CONFIDENTIALITY	O.INTEGRITY	O.NONREPUDIATION	O.ACCESSCONTROL	O.ACCOUNTABILITY	O.ALGORITHM
FAU_GEN.1					X	
FAU_GEN.2					X	
FCS_CKM.1/LK	X	X				X
FCS_CKM.1/SEK	X	X				X
FCS_CKM.5/PBKDF2	X	X				X
FCS_CKM.5/HKDF	X	X				X
FCS_COP.1/hash		X	X			X
FCS_COP.1/ECDH	X	X				X
FCS_COP.1/AES	X	X				X
FCS_COP.1/ECDSA		X	X			X
FDP_ITC.1/AccountKeyringImport	X	X	X			
FDP_IFC.1/AccountKeyringImport	X	X				
FDP_IFF.1/AccountKeyringImport	X	X				
FDP_ITC.2/PeerPublicKeyringImport	X	X	X			
FDP_IFC.1/PeerPublicKeyringImport	X	X				
FDP_IFF.1/PeerPublicKeyringImport	X	X				
FDP_ITC.2/RoomBlockchainImport	X	X	X			
FDP_IFC.1/RoomBlockchainImport	X	X				
FDP_IFF.1/RoomBlockchainImport	X	X				
FDP_ITC.2/CreateRoomBlock	X	X	X			
FDP_IFC.1/CreateRoomBlock	X	X		X		
FDP_IFF.1/CreateRoomBlock	X	X		X		
FDP_DAU.2/CreateRoomBlock			X			
FDP_ITC.2/OwnerRequestImport	X	X	X			
FDP_IFC.1/OwnerRequestImport	X	X				
FDP_IFF.1/OwnerRequestImport	X	X				
FDP_ITC.2/PeerRequestImport	X	X	X			
FDP_IFC.1/PeerRequestImport	X	X				
FDP_IFF.1/PeerRequestImport	X	X				
FDP_ITC.2/AnonymousRequestImport	X	X	X			
FDP_IFC.1/AnonymousRequestImport	X	X				
FDP_IFF.1/AnonymousRequestImport	X	X				
FDP_IFC.1/EncryptResponse	X	X				
FDP_IFF.1/EncryptResponse	X	X				
FDP_DAU.2/EncryptResponse			X			
FDP_ACC.1/AdminRequest				X		
FDP_ACF.1/AdminRequest				X		
FDP_ACC.1/MutatingRequest				X		
FDP_ACF.1/MutatingRequest				X		
FDP_ACC.1/Drivefileoperation				X		
FDP_ACF.1/Drivefileoperation				X		

Security Functional Requirement	O.CONFIDENTIALITY	O.INTEGRITY	O.NONREPUDIATION	O.ACCESSCONTROL	O.ACCOUNTABILITY	O.ALGORITHM
FIA_UID.1					X	
FIA_UAU.1					X	
FMT_MSA.1	X	X				
FMT_MSA.3	X	X				
FMT_SMF.1					X	
FMT_SMR.1					X	
FPT_TDC.1		X				
FTP_ITC.1/P2P	X	X				
FTP_ITC.1/Server	X	X				

Table 18: SFRs mapping to Security Objectives

6.3.2.Security functional requirements sufficiency

The following rationale provides justification for each security objective for the TOE, showing that the security functional requirements are suitable to meet and achieve the security objectives.

Security Objective	Rationale
O.CONFIDENTIALITY	The objective: • The TOE shall provide mechanisms that protect the information of a transmitted data file such that its content is confidentiality-protected and only accessible for authorized users. The confidentiality of cryptographic keys stored on the TOE should also be protected. Is met by: • FCS_CKM.1/* which ensure trusted key generation for confidentiality protection. • FCS_CKM.5/* which ensure trusted key derivation for confidentiality protection. • FCS_COP.1/ECDH and FCS_COP.1/AES a which provide trusted cryptographic algorithms and key lengths for confidentiality protection. • FDP_IFC.1/* and FDP_IFF.1/* which ensure that the TOE enforce confidentiality protected import SFPs when importing user data, controlled under the SFP, from outside of the TOE. • FDP_ITC.1/* and FDP_ITC.2/* which provides confidentiality protected user data import. • FMT_MSA.1 which ensures trusted management of security attributes for confidentiality protection. • FMT_MSA.3 which ensures that nobody can specify alternative initial values to override the default values when an object or information is created for confidentiality protection. • FTP_ITC.1/* which ensures confidentiality protected trusted channel for import of requests, keyrings and room blockchain.

O.INTEGRITY	The objective: - The TOE shall provide mechanisms that detect if an attacker has tampered with a transmitted data file (i.e. replacing or modifying the content of the data file). The integrity of cryptographic keys stored on the TOE should also be protected. Is met by: - FCS_CKM.1/* which ensure trusted key generation for integrity protection. - FCS_CKM.5/* which ensure trusted key derivation for integrity protection. - FCS_COP.1/* which provide trusted cryptographic algorithms and key lengths for integrity protection. - FDP_IFC.1/* and FDP_IFF.1/* which ensure that the TOE enforce Integrity protected import SFPs when importing user data, controlled under the SFP, from outside of the TOE. - FDP_ITC.1/* and FDP_ITC.2/* which provides integrity protected user data import. - FMT_MSA.1 which ensures trusted management of security attributes for integrity protection. - FMT_MSA.3 which ensures that nobody can specify alternative initial values to override the default values when an object or information is created for integrity protection. - FPT_TDC.1 which ensures consistent interpretation of data import from Clarabot Server. - FTP_ITC.1/* which ensures confidentiality protected trusted channel for import of requests, keyrings and room blockchain.
O.NONREPUDIATION	The objective: - The TOE shall provide provide a capability to generate evidence that can be used as a guarantee of the validity of the Room's blockchain and responses. Is met by: - FCS_COP.1/hash and FCS_COP.1/ECDSA which provide trusted cryptographic algorithms and key lengths nonrepudiation (signature creation). - FDP_ITC.1/* and FDP_ITC.2/* which ensure nonrepudiation of user data imported by the TOE. - FDP_DAU.2/EncryptResponse which ensures nonrepudiation of the TOE responses. - FDP_DAU.2/CreateRoomBlock ensures nonrepudiation of of room blockchain.
O.ACCESSCONTROL	The objective: - The TOE shall ensure that TOE resources are shared only on the basis of (owner and Room Administrator)-defined access permissions. Is met by: - FDP_ACC.1/* and FDP_AFC.1/* which ensure that the TOE enforce security attribute based access control SFPs. - FDP_IFC.1/CreateRoomBlock and FDP_IFF.1/CreateRoomBlock ensures that only legitimate blockchain data with permissions are imported to the TOE.
O.ACCOUNTABILITY	The objective: - The TOE shall create audit records for security-relevant events, recording the event details and the subject associated with the event.

	Is met by: • FMT_SMR.1 which ensures that different security roles are defined and that the users are associated with roles. • FIA_UID.1 and FIA_UAU.1 enforce identification and authentication of users. • FAU_GEN.1 and FAU_GEN.2 which provide audit data generation and associate each auditable event with the identity of the user that caused the event. • FMT_SMF.1 which ensures different management functions, including “View audit records”.
O.ALGORITHM	The objective: • The TOE shall only allow the use of cryptographic algorithms and key lengths recommended by a standard, authoritative document should be used. Is met by: • FCS_CKM.1/* which ensure that standard key generation is used. • FCS_CKM.5/* which ensure that standard key derivation is used. • FCS_COP.1/* which ensure that standard cryptographic algorithms and key lengths are used.

Table 19: Security Objectives mapping to SFRs

6.3.3. Satisfaction of SFR dependencies

The dependencies between SFRs are addressed as shown in the following table. Where a dependency is not met in the manner defined in [CC2] then a rationale is provided for why the dependency is unnecessary or else met in some other way.

SFR	Dependencies	Fulfilled by
FAU_GEN.1	FPT_STM.1	see Note_1
FAU_GEN.2	FAU_GEN.1 FIA_UID.1	FAU_GEN.1 FIA_UID.1
FCS_CKM.1/LK FCS_CKM.1/SEK	[FCS_CKM.2 or FCS_CKM.5 or FCS_COP.1]	FCS_CKM.5/HKDF
	FCS_CKM.3	see Note_2
	[FCS_RBG.1 or FCS_RNG.1]	see Note_3
	FCS_CKM.6	see Note_4
FCS_COP.1/hash	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1 or FCS_CKM.5]	hash function does not require key
	FCS_CKM.3	
FCS_COP.1/ECDH	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1 or FCS_CKM.5]	CSK: FDP_ITC.1/AccountKeyringImport CPK: FDP_ITC.2/PeerPublicKeyringImport APK: FDP_ITC.2/AnonymousRequest Import
	FCS_CKM.3	see Note_2
FCS_COP.1/AES	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1 or FCS_CKM.5]	FCS_CKM.5/PBKDF2 FCS_CKM.5/HKDF
	FCS_CKM.3	see Note_2
FCS_COP.1/ECDSA	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1 or FCS_CKM.5]	sign: FDP_ITC.1/AccountKeyringImport validate: FDP_ITC.2/PeerPublicKeyring Import
	FCS_CKM.3	see Note_2
FCS_CKM.5/PBKDF2	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1/AES

SFR	Dependencies	Fulfilled by
	FCS_CKM.6	see Note_4
FCS_CKM.5/HKDF	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1/AES
	FCS_CKM.6	see Note_4
FDP_ITC.1/ AccountKeyringImport	[FDP_ACC.1 or FDP_IFC.1] FMT_MSA.3	FDP_IFC.1/AccountKeyringImport FMT_MSA.3
FDP_IFC.1/ AccountKeyringImport	FDP_IFF.1	FDP_IFF.1/AccountKeyringImport
FDP_IFF.1/ AccountKeyringImport	FDP_IFC.1	FDP_IFC.1/AccountKeyringImport
	FMT_MSA.3	FMT_MSA.3
FDP_ITC.2/ PeerPublicKeyringImport	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.1/PeerPublicKeyringImport
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1/Server
	FPT_TDC.1	FPT_TDC.1
FDP_IFC.1/ PeerPublicKeyringImport	FDP_IFF.1	FDP_IFF.1/PeerPublicKeyringImport
FDP_IFF.1/ PeerPublicKeyringImport	FDP_IFC.1	FDP_IFC.1/PeerPublicKeyringImport
	FMT_MSA.3	FMT_MSA.3
FDP_ITC.2/ RoomBlockchainImport	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.1/RoomBlockchainImport
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1/Server
	FPT_TDC.1	FPT_TDC.1
FDP_IFC.1/ RoomBlockchainImport	FDP_IFF.1	FDP_IFF.1/RoomBlockchainImport
FDP_IFF.1/ RoomBlockchainImport	FDP_IFC.1	FDP_IFC.1/RoomBlockchainImport
	FMT_MSA.3	FMT_MSA.3
FDP_ITC.2/ CreateRoomBlock	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.1/CreateRoomBlock
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1/Server
	FPT_TDC.1	FPT_TDC.1
FDP_IFC.1/CreateRoomBlock	FDP_IFF.1	FDP_IFF.1/CreateRoomBlock
FDP_IFF.1/ CreatRoomBlock	FDP_IFC.1	FDP_IFC.1/CreateRoomBlock
	FMT_MSA.3	FMT_MSA.3
FDP_ITC.2/OwnerRequestImport	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.1/OwnerRequestImport
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1/P2P
	FPT_TDC.1	FPT_TDC.1
FDP_IFC.1/OwnerRequestImport	FDP_IFF.1	FDP_IFF.1/OwnerRequestImport
FDP_IFF.1/OwnerRequestImport	FDP_IFC.1	FDP_IFC.1/OwnerRequestImport
	FMT_MSA.3	FMT_MSA.3
FDP_ITC.2/PeerRequestImport	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.1/PeerRequestImport
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1/P2P
	FPT_TDC.1	FPT_TDC.1
FDP_IFC.1/PeerRequestImport	FDP_IFF.1	FDP_IFF.1/PeerRequestImport
FDP_IFF.1/PeerRequestImport	FDP_IFC.1	FDP_IFC.1/PeerRequestImport
	FMT_MSA.3	FMT_MSA.3
FDP_ITC.2/ AnonymousRequestImport	[FDP_ACC.1 or FDP_IFC.1]	FDP_IFC.1/AnonymousRequestImport
	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC.1/P2P
	FPT_TDC.1	FPT_TDC.1
FDP_IFC.1/ AnonymousRequestImport	FDP_IFF.1	FDP_IFF.1/AnonymousRequestImport
FDP_IFF.1/ AnonymousRequestImport	FDP_IFC.1	FDP_IFC.1/AnonymousRequestImport
	FMT_MSA.3	FMT_MSA.3
FDP_IFC.1/EncryptResponse	FDP_IFF.1	FDP_IFF.1/EncryptResponse
FDP_IFF.1/EncryptResponse	FDP_IFC.1	FDP_IFC.1/EncryptResponse
	FMT_MSA.3	FMT_MSA.3
FDP_DAU.2/EncryptResponse	FIA_UID.1	FIA_UID.1
FDP_ACC.1/AdminRequest	FDP_ACF.1	FDP_ACF.1/AdminRequest

SFR	Dependencies	Fulfilled by
FDP_ACF.1/AdminRequest	FDP_ACC.1	FDP_ACC.1/AdminRequest
	FMT_MSA.3	FMT_MSA.3
FDP_ACC.1/MutatingRequest	FDP_ACF.1	FDP_ACF.1/MutatingRequest
FDP_ACF.1/MutatingRequest	FDP_ACC.1	FDP_ACC.1/MutatingRequest
	FMT_MSA.3	FMT_MSA.3
FDP_ACC.1/Drivefileoperation	FDP_ACF.1	FDP_ACF.1/Drivefileoperation
FDP_ACF.1/Drivefileoperation	FDP_ACC.1	FDP_ACC.1/Drivefileoperation
	FMT_MSA.3	FMT_MSA.3
FIA_UID.1	-	n/a
FIA_UAU.1	FIA_UID.1	FIA_UID.1
FMT_MSA.1	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1/* FDP_IFC.1/*
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.3	FMT_MSA.1	FMT_MSA.1
	FMT_SMR.1	FMT_SMR.1
FMT_SMF.1	-	n/a
FMT_SMR.1	FIA_UID.1	FIA_UID.1
FPT_TDC.1	-	n/a
FTP_ITC.1/P2P	-	n/a
FTP_ITC.1/Server	-	n/a

Table 20: Satisfaction of dependencies for functional requirements

Note_1 FPT_STM.1 Reliable time stamps isn't included since according to OE.TIME the execution platform provides a reliable time source.

Note_2 FCS_CKM.3 Cryptographic Key Access isn't included since the TOE does not use the generated keys outside the TOE (e.g. backup, archival, escrow, recovery).

Note_3 FCS_RBG.1 Random bit generation and FCS_RNG.1 Random number generation aren't included since the TOE rely on underlying platform for random number generation (see: OE.CSPRNG)

Note_4 FCS_CKM.6 Cryptographic Key Destruction isn't included since the TOE as a software application rely on underlying platform for memory and storage management.

6.3.4. Satisfaction of SAR dependencies

SAR	Dependencies	Satisfied by
EAL2 package	(dependencies of EAL2 package are not reproduced here)	By construction, all dependencies are satisfied in a CC EAL package

Table 21: Satisfaction of dependencies for assurance requirements

6.3.5. Rationale for chosen security assurance requirements

The evaluation assurance level for this ST is EAL2.

EAL2 is applicable, since the developers require a low to moderate level of independently assured security.

7. TOE summary specification

7.1.Owner identification and authentication

The identification and logon mechanism assumes that the Owner already registered an account in Calarabot Server.

- The Owner types the email and password in the login form of the TOE.
- The TOE sends the email part of the login form to the Clarabot Server.
- Clarabot Server returns the computed salt by the stored salt (established at registration)
    - If the email is unknown Clarabot Server still returns a deterministic computed salt for the unregistered email that is indistinguishable from valid salts. This prevents the discovery of registered email addresses.
- The TOE uses the password and the salt returned from the Clarabot Server, and derives two keys using PBKDF2:
    - Authentication Key (AK)
    - Encryption Key (EK)
- Using CSPRNG TOE generates SEK1 and SEK2 of length 16 bytes and SEK3 of 32 bytes.
- SEK1 and SEK3 will be stored by the TOE for later use.
- SEK2 and AK will be send to Clarabot Server.
- Clarabot Server authenticates the user based on the hashed AK stored for the account during registration.
- If the authentication succeeded, Clarabot Server stores SEK2 for the account.

The Owner identification and authentication security function is designed to satisfy the following security functional requirements:

- FIA_UID.1 – identification based on email address
- FIA_UAU.1 – authentication based on password and optional 2FA secret
- FCS_CKM.5/PBKDF2 – AK and EK derivation from password

7.2.Initial importing and local storing of the Owner's account keyring

The Owner's account keyring is imported after a successful authentication of the Owner.

- Identification and authentication: see: 7.1.
- If the authentication succeeded, Clarabot Server returns
    - account id
    - the Owner's account keyring containing
        - master-key (MK) encrypted by EK
        - ed25519-secret-key (ESK) encrypted by MK
        - curve25519-secret-key (CSK) encrypted by MK
    - the server's ephemeral curve25519-public-key (SECPK)
    - a random session-id encrypted by a derived key from the shared secret of SECPK and CSK
- The MK will be decrypted using with a HKDF expanded key of 64 bytes from the EK (Private Key Cryptographic Context)
- MK will be expanded with HKDF to 64 bytes

- expanded MK can be used to decrypt ed25519-secret-key (ESK) and curve25519-secret-key (CSK) from the Owner's account keyring (Account Root Key Cryptographic Context)
- decrypted CSK can be used to decrypt the session-id by a HKDF expanded key of 64 bytes from the shared secret of CSK and SECPK (Login Challenge Cryptographic Context)
- session-id is stored in local store (Local Storage Cryptographic Context)
- the account keyring is stored in local store encrypted by a HKDF expanded key of 64 bytes from SEK1, SEK2 and SEK3 (Local Storage Cryptographic Context)

The security function is designed to the following security functional requirements:

- FDP_ITC.1/AccountKeyringImport – Owner's encrypted keyring is imported and decrypted
- FDP_IFC.1/AccountKeyringImport and FDP_IFF.1/AccountKeyringImport – The AES-SIV authentication result is checked on the imported keyring
- FCS_CKM.5/PBKDF2 – derive AK and EK from password
- FCS_CKM.5/HKDF – expand MK
- FCS_COP.1/AES – decrypt MK using EK
- FCS_COP.1/AES – decrypt ESK and CSK (using MK), decrypt Session Key (using CSK)
- FCS_CKM.1/SEK – generate SEK1, SEK2, SEK3

7.3.Subsequent access to the Owner's account keyring

During the session (until the session-id is available) Owner's account keyring can be decrypted automatically without entering the password:

- session-id is retrieved from local store,
- SEK1 and SEK3 are retrieved from local store,
- SEK2 is retrieved from Clarabot Server,
- Session Encryption Key (SEK) derived from SEK1, SEK2 and SEK3 using HKDF,
- MK, ESK and CSK can be decrypted using SEK from local store. (Local Storage Cryptographic Context)

If the session has expired (the valid session-id is not available) the local stored Owner's account keyring can not be decrypted.
A new identification/authentication/session creation will be required.
This requires user intervention, entering the password (see 7.1)

The security function is designed to the following security functional requirements:

- FCS_CKM.5/HKDF – derive SEK (from SEK1, SEK2 and SEK3)
- FCS_COP.1/AES – decrypt MK, ESK, and CSK (using SEK)

7.4.Decrypt incoming request

The context of the request contains (beside other fields):

- requesting account id
- Room id

If the requesting account id is the Owner:

- a derived master key (DMK) is created using HKDF from MK and with salt combined from static salt, a time-window value from the current timestamp and keyring id. (Nano Own Cryptographic Context)
- the encrypted request is decrypted using algorithm AES-SIV with key DMK (Nano Own Cryptographic Context)

If the requesting account id is a peer (account id is present, but is not the Owner's account id):

- the trust of the account id is established (see 7.5) and the public keyring of the peer (including CPK, the public part of the peer's Curve25519 key) is retrieved
- the Room's pinned key is retrieved from Clarabot server (if not already available) (Account Room Key Context)
- a derived peer key (DPK) is created (Nano Cryptographic Context)
  - using ECDH with the secret Owner's CSK and the public peer's CPK
  - deriving the result with HKDF using a salt combined from a static value, a time-window value from the current timestamp, the ordered account ids of the peers and the Room's pinned key
- the encrypted request is decrypted using algorithm AES-SIV with key DPK (Nano Cryptographic Context)

If the requesting account id is an anonymous user (account id is not present, and anonymous access is allowed):

- the public part of the anonymous ephemeral Curve25519 key (APK) is part of the request
- the Room's pinned key is retrieved from Clarabot server (if not already available) (Account Room Key Context)
- a derived peer key (DPK) is created (Nano Cryptographic Context)
  - using ECDH with the secret Owner's CSK and the public APK
  - deriving the result with HKDF using a salt combined from a static value, a time-window value from the current timestamp, the ordered account ids of the peers and the Room's pinned key
- the encrypted request is decrypted using algorithm AES-SIV with key DPK

The security function is designed to the following security functional requirements:

- FDP_ITC.2/*RequestImport – Owner's, Peer's and Anonymous's requests are imported and decrypted
- FDP_IFC.1/*RequestImport and FDP_IFF.1/*RequestImport – The AES-SIV authentication result is checked on the imported request

- FDP_IFC.1/AnonymousRequestImport and FDP_IFF.1/AnonymousRequestImport – The anonymous access flag is checked on the imported request
- FCS_CKM.5/HKDF – key derivation operations described above
- FCS_COP.1/AES – decryption operations described above
- FCS_COP.1/ECDH – key agreement operations described above

7.5.Determining trust for incoming request

The metadata of the request contains:

- requesting account id

Determination of the trust of the requesting account id is needed for peer account ids (account id is present, but is not the Owner's account id)

If the requesting account id has never been trusted before:

- the public keyring of the account id is retrieved from Clarabot Server, containing:
    - public part of account's Curve25519 key (CPK)
    - signature of CPK signed by secret part of the account's Ed25519 key (ESK)
    - public part of the account's Ed25519 key EPK (trust root)
    - trust fingerprint (explicit trust by the Owner)
- The trust fingerprint is a result of explicit trust of between the Peer and the Owner outside the scope of the TOE: (Account Trust Cryptographic Context)
    - the Owner selects explicit trust of the EPK of the Peer in the web application
    - A data (DATA_TRUST) is derived (using HKDF) from Owner's MK and Peer's EPK
    - A key (KEY_TRUST) is derived (using HKDF) from Owner's MK and Peer's EPK
    - Trust fingerprint (TF) is calculated as AES-SIV-N0 (KEY_TRUST, DATA_TRUST)
- the signature of the CPK is checked for the trust to be established (Account Signature Cryptographic Context)
- the public part of account's Curve25519 key is used for the ECDH in deriving DPK, and the request is using an authenticated cipher (AES-SIV), so the trust chain can be built from the encrypted request to the public part of the account's Ed25519 key (trust root)
- in the evaluated version the public keyring must contain a trust fingerprint. The fingerprint is calculated in the TOE and checked against the trust fingerprint in the public keyring after it gets successfully decrypted. This mechanism is called explicit trust. (Account Trust Cryptographic Context)

The security function is designed to the following security functional requirements:

- FDP_ITC.2/PeerPublicKeyringImport - Peer's public keyring is imported
- FDP_IFC.1/PeerPublicKeyringImport and FDP_IFF.1/PeerPublicKeyringImport - The trust fingerprint is checked on the imported request
- FCS_COP.1/ECDSA – signature validations described above
- FCS_COP.1/ECDH – key agreement operations described above
- FCS_COP.1/AES – decryption operations described above
- FCS_CKM.5/HKDF – key derivation operations described above

7.6.Handling admin requests

Admin requests are requests of the Owner, that can affect the configuration of the local Nano instance.

Admin requests can be determined by the type of the request.

Admin commands will be handled based on the security configuration "remote_admin_policy":

- allow: all admin commands will be accepted
- restrict: admin commands will be accepted if the request contains admin password that matches the admin password set on the Nano instance
- deny: all admin commands will be rejected

The security function is designed to the following security functional requirements:

- FDP_ACC.1/AdminRequest, FDP_ACF.1/AdminRequest – perform access control based on remote_admin_policy configuration parameter and the admin password

7.7.Handling mutating requests

Mutating handlers are request handlers that change the state of the local Nano instance.

Mutating requests can be determined by the request type.

If the request is a mutating request, then the request must contain an unique session token, that has not been used before (the TOE keeps track of the currently valid tokens)

The requestor will ask for a new session token before the mutating request using the "create_mutation_id" request.

The request will be rejected if it does not contain a valid session token (or a session token is used more than once).

The security function is designed to the following security functional requirements:

- FDP_ACC.1/MutatingRequest, FDP_ACF.1/MutatingRequest – perform access control for requests that would result in a state change on the TOE, based on session token uniqueness check

7.8. Enforce access controls on incoming request

The permissions of a Room that the TOE is attached to are stored in Room blocks.

The Room blocks are stored in both the TOE and the Clarabot Server a linked directed acyclic graph. Each Room block is linked to the previous block by hashing (SHA2-512) and is signed by the Owners ESK.

A Room block contains the following security attributes:

- membership flags
    - member id and associated flags where flags are
        - active, active negate
        - administrator, administrator negate
        - Drive write, Drive write negate
        - Drive adhoc, Drive adhoc negate
        - Drive upload, Drive upload negate
- group flags
    - associated flags where flags are
        - allow anonymous, allow anonymous negate

When the TOE receives a CreateRoomBlock request, it creates a new Room block from the request data signs the block (Account Signature Cryptographic Context) and adds the new block to the chain.

On each startup of the TOE, or in case of a push notification originating from Clarabot Server on a change in the Room permissions, the TOE downloads the new Room blocks from Clarabot Server and stores the latest block hash in the configuration.

The downloaded chain is verified (both the signatures and the hash links), omission of blocks is detected by checking the latest block hash stored in the TOE configuration.

On a new request the chain is evaluated, the flags for the requestor's account can be:

- active
- administrator
- Drive write
- Drive adhoc
- Drive upload

The request type determines the required flags for the request to be handled. If the requestor's account does not have the required flags, then the request will be rejected.

If the group config has no allow anonymous, and the requestor is anonymous then the request will be rejected. Additionally if the Owner sets the deny anonymous setting in the TOE, the anonymous requests are rejected regardless of the group config.

Creating new block has one added function: when a new block is created, delta keys are created by encrypting the RK using the owner's CSK and the new Peer's CPK. (Room Key Cryptographic Context)

These delta keys are also uploaded to the server, so the new peer's can retrieve the encrypted RK.

On creating new block the graph can be optimized by the TOE using aggregate blocks:

- it is checked that the current block-chain is valid
- it is checked that a there are at least 48 blocks in the graph older than 21 days and 2 hours
- in this case a new replacement aggregate block is created (by aggregating all security attributes in the old blocks) creating a new graph

The security function is designed to the following security functional requirements:

- FDP_ITC.2/RoomBlockChainImport - import room blockchain from Clarabot Server
- FDP_IFC.1/RoomBlockChainImport and FDP_IFF.1/RoomBlockChainImport - the current block chain is validated on import
- FCS_COP.1/hash – hashing latest block, create hash values for signature validations
- FCS_COP.1/ECDSA – sign the latest block using Owner's ESK
- FCS_COP.1/AES – encrypt RK using the new Peer's CPK (during create a delta key).

7.9.Enforcing Drive configuration on handling file requests

For requests that operate on the attached Drive, the TOE enforces that the selected file/path must be included in the Drive.

The filesystem access of drive operations are strictly validated according to the rules of the current platform for each path and all its components. Before any operation takes place, the path is built up by its components one by one from the drive-root. As the target path is resolved by the OS API gradually including all kinds of symbolic links, checking drive-containment on each step ensures that all operations will be inside the designated directory of the drive.

The TOE is designed to be indifferent to the storage backend and filesystem to support all platforms potentially. This is made possible by only utilizing the generally available POSIX filesystem API and its equivalent for Windows. This also means that the platform specific advanced security mechanisms are out of scope for the TOE to manage.

The security function is designed to the following security functional requirements:

- FDP_ACC.1/DriveFileOperation and FDP_ACF.1/DriveFileOperation – the requested file/path is validated and must be in the Drive

7.10. Encrypt response

After the handler processed the response, it must be encrypted for the requestor

- the request handler will decide dynamically if the request is cacheable or not
- a request key is created from source data (depends on source data, file path, modification timestamp and size)
- the serialized request key is created from the request key using serialization and hashing (SHA3-256), the key has a maximum length enforced by the server, only the tail of the key is hashed so the digest will replace its input resulting in a key of maximum length exactly
- transfer key/cache key is created that uniquely identifies the state of the resource that is the subject of the request without revealing any information about it by encrypting the serialized request key (SRK) with AES-SIV using a local secret that is only available on the TOE:
    - the local secret is SERVER_TRANSFER_KEY_ENCRYPTION if the response is non-cacheable
    - the local secret is SERVER_CACHE_KEY_ENCRYPTION if the response is cacheable
    - for SIV a deterministic salt (nonce) is created using HKDF from the SERVER_REQUEST_KEY_IKM local secret and the SHA256 digest of the SRK, additionally for non-cacheable responses the head of the request is added to the digest so the encrypted transfer key will be unique for virtually each request
- a part key (PK) is generated (Nano Data Cryptographic Context) with HKDF when the response is not a single part or is cacheable by using either
    - the generated cache key and the local secret SERVER_CACHE_KEY_IKM if the response is cacheable
    - the generated transfer key and the local secret SERVER_TRANSFER_KEY_IKM if the response is not cacheable

The response will always be made up of the following parts

- setup part
- 0..n response chunks

The setup part contains:

- P2P encrypted
    - part key (PK) (if any, only present if there are chunks after the setup part)
    - request authenticity hash (RAH) – this is a truncated SHA256 digest of the cache or transfer key and the head of the request, this cryptographically bind together the request with the response in a way that is easy and secure to verify by the requestor, this prevents response-replay attacks by the server because each request is always encrypted and are always unique
- the number of chunks

- transfer key/cache key

The setup part is always encrypted in the same way the requestor used (see encrypt incoming request) (Nano Cryptographic Context or Nano Own Cryptographic Context)

Subsequent chunks are bound to a setup part by the transfer/cache key (can be included in the request). The cache key enables the server to cache and serve the chunks to a peer, after the cache key and the part key was retrieved from the setup part

Each chunk is encrypted by PK using AES-SIV. (Nano Data Cryptographic Context)

SHA512 hash with CSPRNG prefix is calculated for cacheable chunk ranges that are always signed by the Owner's ESK. These fingerprints are placed in random chunks while the last chunk always contains one for the last range of chunks. These signed fingerprints of ranges always cover all the chunks of a data transfer.
Not cacheable, but multipart responses do not need to have digital signatures by the Owner, because in this case the PK is exclusively only disclosed to the requesting party.

The security function is designed to the following security functional requirements:

- FDP_IFC.1/EncryptResponse and FDP_IFF.1/EncryptResponse – TK or CK is used for PK generation based on the value of the cache flag
- FDP_DAU.2/EncryptResonse – the requestor Peer when successful decrypts response also verifies the validity/integrity of the response and the identity of the Owner.
- FCS_CKM.1/LK -
- FCS_COP.1/AES – encrypt setup part (using key exchanged for incoming request), encrypt chunks (using PK)
- FCS_COP.1/ECDSA – sign HMAC using Owner's ESK
- FCS_COP.1/hash – for create SRK (serialized request key) (SHA3-256)

7.11. Cryptographic support

CNC uses cryptography to

- decrypt and validate the integrity of the requests retrieved from Clarabot server
- encrypt and protect the integrity the responses sent through Clarabot server
- validate the trust for accounts
- protect the stored application data

For the decription of the cryptographic keys see Table 1.

The TOE uses cryptographic contexts for common scenarios. Table 22 describes the usage of cryptographic contexts.

Name	Purpose
Account Room Key	Encryption for account's room key Algorithm(s): AES-SIV-N0, HKDF-SHA256, Keys: Key derived from MK Salt inputs: Owner account id, Room id, version of the room key, version of the context, static salt, custom prefix V1
Local Storage	Encryption for locally stored sensitive data Algorithm(s): AES-SIV-N96, HKDF-SHA512, HKDF-SHA256 Keys: SEK1, SEK2, SEK3 V1 static salt is used in HKDF Uses HKDF-SHA256 V2 dynamic salt is used in HKDF Uses HKDF-SHA512
Account Root Key	Encryption of ESK, CSK in Owner's account keyring Algorithm(s): AES-SIV-N0, HKDF-SHA256, Keys: Key derived from MK Salt inputs: Owner account id, version of the secret key, version of the context, static salt, custom prefix V1
Account Signature	Signature using Owner's ESK Signature verification using Owner's EPK Algorithm(s): Ed25519 Keys: ESK, EPK V1
Account Trust	Create and validate encrypted trust fingerprint of a peer account Algorithm(s): AES-SIV-N0, HKDF-SHA256 Keys: Key derived from MK

Name	Purpose
	Salt inputs: Owner account id, version of the context, static salt, custom prefix, peer account id V2 Fingerprint is made by HKDF derived from MK with a custom prefix and the peer trust root
Login Challenge	Encryption for login challenge (session-identifier) Algorithm(s): AES-SIV-N96, HKDF-SHA256, ECDH Keys: Owner's CSK, SECPK V1
Nano	Decryption for Nano request payload and encryption of setup response payload when requesting account is not the Owner. Algorithm(s): AES-SIV-N96, HKDF-SHA256, ECDH Keys: Key derived from Owner's CSK and Peer's CPK with ECDH Salt inputs: Ordered ids of the Owner and requester, RK, version of the context, static salt, time-window value from current timestamp, custom prefix V1
Nano Data	Encryption for Nano response payload Algorithm(s): AES-SIV-N96 Keys: PK V1
Nano Own	Encryption for Nano response payload and encryption of setup response payload where the requestor is the Owner Algorithm(s): AES-SIV-N96, HKDF-SHA256, Keys: Key derived from MK Salt inputs: Owner account id, version of the context, static salt, time-window value from current timestamp, custom prefix V1
Private Key	Encryption for MK Algorithm(s): AES-SIV-N0, HKDF-SHA256, Keys: EK Salt inputs: version of MK, version of the context, static salt, custom prefix V1
Room Key	Encryption of a RK for a Peer when granting room access by config blockchain Algorithm(s): AES-SIV-N0, HKDF-SHA256, ECDH Keys: Key derived from Owner's CSK and Peer's CPK with ECDH Salt inputs: Ordered ids of the Owner and requester, id of the room, version of the RK, version of the context, static salt, custom prefix

Name	Purpose
	V1

Table 22: The usage of cryptographic contexts

The Table 23 describes the usage of cryptographic algorithm:

Function	Purpose
AES-SIV-N0	Deterministic data encryption/decryption for key-wrapping and securing high-entropy or unique values
AES-SIV-N96	Data encryption/decryption using a nonce with the length of 12 bytes
Ed25519 Signature	Data signature/authentication
PBKDF2	Key derivation from password
ECDH	Key agreement for P2P encryption
HKDF-SHA256	Key derivation function from salt and KIM (Key Input Material)
HKDF-SHA512	Key derivation function from salt and KIM (Key Input Material)
SHA3-256	Data hashing
SHA2-256	Data hashing truncated to 128 bits digest to prevent length-extension attacks, when needed
SHA2-512	Data hashing truncated to 256 bits digest to prevent length-extension attacks, when needed

Table 23: The usage of cryptographic algorithm

All the cryptographic functions above are standard cryptographic functions.

The Cryptographic support security function is designed to satisfy the following security functional requirements:

- FCS_COP.1/hash
- FCS_COP.1/ECDH
- FCS_COP.1/AES
- FCS_COP.1/ECDSA
- FCS_CKM.5/PBKDF2
- FCS_CKM.5/HKDF
  the table above shows the utilized cryptographic algorithms that match the requirements

7.12. Security management

The Owner can use the GUI of the Nano Client to invoke the security management functions of the TOE.

The available security related managment functions are:

- Change password
- Reset Nano Client
- Set admin password
- Create Drive
- Delete drive
- Attach room to drive manually
- View audit records

Some management functions are invoked on handling specific requests only from the Owner:

- Change admin password
- Update deny anonymous
- Update require explicit peer trust
- Remote drive management

Some management functions are invoked on handling specific requests from a Room Administrator or Owner:

- Create Room block

The Security management security function is designed to satisfy the following security functional requirements:

- FMT_SMF.1 – CNC supports the required management functions
- FMT_SMR.1 – CNC restricts management functions to the roles described above
- FDP_ITC.2/CreateRoomBlock, FDP_IFC.1/CreateRoomBlock and FDP_IFF.1/CreateRoomBlock - only RoomAdministator or Owner can send the CreateRoomBlock request
- FDP_DAU.2/CreateRoomBlock – the created room block is signed by Owner's ESK

7.13. Security audit

The TSF creates an audit log that generates among other things the following events:

- Remote access to Drive
- Nano remote configuration queried
- Configuration changes
- Room management operations

The audit events are stored in file, and the access-controls of the OS restrict modification or deletion of the audit records by unauthorized users.

The Security audit security function is designed to satisfy the following security functional requirements:

- FAU_GEN.1 - CNC stores the required audit record in text format.
- FAU_GEN.2 - For audit records where it is applicable, the account id of the user who initiated the operation is also recorded

7.14. View audit log

The Owner can access a management function to view audit log on the UI.

The function opens the audit log as a text file with the default text viewer application of the computer the CNC is installed on:

- FMT_SMF.1 – This security function implements the required view audit log management function

8. References and Acronyms

8.1.References

[CC1] ISO/IEC 15408-1:2022 Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

1: Introduction and general model

[CC2] ISO/IEC 15408-2:2022 Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

Part 2: Security functional components

[CC3] ISO/IEC 15408-3:2022 Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

Part 3: Security assurance components

[CC4] ISO/IEC 15408-4:2022 Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

4: Framework for the specification of evaluation methods and activities

[CC5] ISO/IEC 15408-5:2022 Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

5: Pre-defined packages of security requirements

[TS 119312] ETSI TS 119312 Electronic Signatures and Infrastructures (ESI); Cryptographic Suites Version 1.4.2 Feb 2022

[FIPS 140-3] FIPS PUB 140-3: Security Requirements for Cryptographic Modules, March 22, 2019

[FIPS 180-4] FIPS PUB 180-4 Secure Hash Standard (SHS), August 2015

[FIPS 186-4] FIPS PUB 186-4: Digital Signature Standard (DSS), July 2013

[FIPS 197] FIPS PUB 197 Advanced Encryption Standard (AES), November 26, 2001

[FIPS 198-1] FIPS PUB 198-1 The Keyed-Hash Message Authentication Code (HMAC), July, 2008

[FIPS 202] FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August, 2015

[PKCS#5] RSA Laboratories - PKCS #5: Password-based Cryptographic Standard, Version 2.1

[RFC 2104] RFC 2104 - HMAC: Keyed-Hashing for Message Authentic

[RFC 5280] RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

[RFC 5297] RFC 5297 - Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)

[RFC 5869] RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function (HKDF)

[RFC 8032] Edwards-Curve Digital Signature Algorithm (EdDSA)

[SP800-38A] NIST Special Publication 800-38A Recommendation for Block Edition Cipher Modes of Operation, December 2001

[SP800-57] NIST Special Publication 800-57 Part 1 Revision 5 Recommendation for Key Management: Part 1 – General

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

[SP800-186] Draft NIST Special Publication 800-186 Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters, October 2019 (Draft)

8.2.Acronyms

AES	Advanced Encryption Standard
AK	Authentication key
APK	Anonymous (Ephemeral) Public Key
ASK	Anonymous (Ephemeral) Secret Key
CC	Common Criteria
CK	Cache key
CNC	Clarabot Nano Client
CNS	Clarabot Nano Server
CPK	curve25519-public-key
CSK	curve25519-secret-key
CSPRNG	Cryptographically Secure Pseudorandom Number Generator
DMK	Derived Master Key
DPK	Derived Peer Key
EAL	Evaluation Assurance Level
ECDH	Elliptic-curve Diffie–Hellman
EK	Encryption Key
EPK	ed25519-public-key
ESK	ed25519-secret-key
HKDF	HMAC-based Key Derivation Function
HMAC	Hash-based Message Authentication Code
LK	Local secret (key)
MK	Master key
PBKDF	Password-based Key Derivation Function
PK	Part key
P2P	peer-to-peer
RK	Room Key
SEK	Session Encryption Key
SHA	Secure Hash Algorithm
SIV	Synthetic Initialization Vector
TK	Transfer key