# Egress Switch Secure Email & File Transfer v4.8 Security Target

| | | |
|---|---|---|
| Client | : | Egress Software Technologies Limited |
| Issue | : | 1.5 |
| Issue date | : | 24 July 2017 |
| Status | : | Master Issue |
| Project code | : | CGI 56133 |
| Document reference | : | 56133.LFL/T280-cons.ST.1 |
| | : | Richard Green (Egress) |
| | | UK Certification Body (NCSC) |
| | | Project File |

| | | | |
|---|---|---|---|
| Prepared by | : | | Ian Mearns (CGI) |
| | | …… | |
| Reviewed by | : | | Richard Green (Egress) |
| | | …… | |
| Agreed | : | | Richard Green (Egress) |
| | | …… | |

# Contents

# 1 Preamble

## 1.1 Document Purpose and Conventions

1. This document is the Security Target (ST) relating to the email encryption functionality of the Egress Switch product ("the Switch") supplied by Egress Software Technologies Limited ("Egress"). It is written to conform to the requirements of the Common Criteria (CC) for Information Technology Security Evaluation (see [CC]).
2. Note that the Switch is a software product (not a hardware communications device); it consists of a number of components that can be configured to provide various encryption-based facilities. The components are installed on underlying platform(s), and rely on these and other platform(s) to supply required services. These platforms constitute the Switch's operational environment (OE); the Switch is the Target of Evaluation (TOE).
3. The precise scope of the TOE and its operational environment that this ST relates to is defined in Sections 2.4.2 and 2.4.3.
4. The "File Transfer" element of the product name refers to the capability to transfer files securely as encrypted email attachments and not the FTP functionality provided by another part of the product.
5. References (see Section 1.2) are given as mnemonics within square brackets. The use of italics for some terminology is explained at the start of the Glossary (Section 1.3).
6. "He" is shorthand for "he or she" (and similarly for "him" and "his"); "administration" is synonymous with "management" (of IT hardware and software); and "email" means "email message and any attachments".
7. The reader is assumed to be familiar with the main terms and concepts used in [CC], and with general IT/crypto terms, e.g. AES, TLS.
8. The structure of this document – see the Contents list above - corresponds closely with that indicated in [CC], Part 1, Section A.2.[1]

## 1.2 References

| Mnemonic | Title/Description | Issuer/File Ref | Version |
|---|---|---|---|
| [CC] | Common Criteria for Information Technology Security Evaluation<br>Part 1: Introduction and General Model<br>Part 2: Security Functional Components<br>Part 3: Security Assurance Components | <br>[CCMB-2012-09-001]<br>[CCMB-2012-09-002]<br>[CCMB-2012-09-003] | V3.1R4, Sept 2012 |
| [CEM] | Common Methodology for Information Technology Security Evaluation, Evaluation methodology | [CCMB-2012-09-004] | V3.1R4, Sept 2012 |

[1] The Extended Components Definition is placed at the end of the document.

## 1.3 Glossary

### 1.3.1 Terms

9. Terms used in this ST which have a specific meaning in the context of threats, objectives or security requirements are italicised in order to emphasise that meaning.

10. Some terms used in the ST taken from [CC] may not be included in the following table; conversely, the definition of some terms below may be slightly different from that given for those terms in [CC]. See also Section 1.1 re general terminology conventions in this ST.

| Term | Meaning |
|---|---|
| *administrator* | An authorised user of the TOE with administrative privileges.[2] |
| *database owner* | A sub-type of *administrator* that can manage the TOE's database. |
| deployment scenario | A number of instances of the TOE that communicate with each other (and possibly with other IT entities) across insecure network(s). |
| ff | and following paragraphs |
| *hacker* | A potential threat agent, either *internal* or *external* (as explained in Section 4.1.2). |
| operational environment | The platform(s) on which the TOE is installed, and which – together with other platform(s) – provide services that the TOE relies on. (Further comments on the distinction between the TOE and its OE are given in Paragraph 47.) |
| *organisation* | An *organisation* (or "tenant") is a conceptual partition of the TOE (covering a subset of the TSF and user data). |
| *organiser* | A sub-type of *administrator* with privileges across an *organisation*. |
| *package* | A *package* contains an encrypted email (and the addresses of the email's sender and recipients), plus, in effect, a link to that email's decryption key.[3] |
| *packreg* | A *packreg* contains the "registration details" for a *package*, including the *package* id, the decryption key for the encrypted email in that *package*, the addresses of the email's sender and recipients, and a link to the relevant *onemail policy* (if any). |
| platform | An item of hardware hosting an operating system and (possibly) other software. |
| *policy* | A *corporate policy* is a set of rules that governs the processing of emails handled by the TOE; it also contains security attributes that are pertinent to the control of access to *packages*. A *corporate policy* applies to the whole TOE or to a single *organisation*; a *onemail policy* applies to a single email, and is linked to that email's *package* via the corresponding *packreg*. |
| *super-user* | A sub-type of *administrator* with privileges across the whole TOE. |

[2] Note that *administrators* and *users* are potential threat agents.

[3] When the TOE creates a *package* it also creates what may be called a *packreg* object, which is stored in the TOE; this *packreg* is what the package is linked to. The *package* itself is sent to each instance of the TOE that deals with one or more of the email recipients. The TOE may also create and store what may be called a *onemail policy* (see *policy*), which is used to specify additional authentication requirements that must be met before the email's decryption key can be released.

| Term | Meaning |
|---|---|
| TOE account | An equivalent concept to an IT operating system user or service account, but implemented by the TOE for TOE users and services. There are different types of TOE account, corresponding to different types of TOE users and services, as explained in Section 7.2.[4] |
| TOE account id | The means of identifying a TOE account.<br>(The id may be referred to as the account username; it has the format user@domain, and in most cases is the same as that user's email address.) |
| *user* | An authorised user of the TOE with no (or very limited) administrative privileges. |
| user (not italicised) | A human or IT entity that interacts with the TOE<br>(as defined in [CC] Part 1). |

### 1.3.2 Abbreviations

11. Some of the following abbreviations are taken from the [CC] Part 1 glossary.

| Acronym | Meaning |
|---|---|
| AD LDS | Active Directory Lightweight Directory Service |
| AES | Advanced Encryption Standard |
| AuS | Authentication Server |
| CBC | Cipher Block Chaining |
| CC | Common Criteria |
| DBMS | Database Management System |
| DBS | Database Server |
| ECP | External Connection Point |
| ESC | Egress Switch Client |
| ESG | Egress Switch Gateway |
| ESI | Egress Switch Infrastructure |
| FIPS | Federal Information Processing Standards (US) |
| FTP | File Transfer Protocol |
| HMAC | Hash-based Message Authentication Code |
| I&A | Identification & Authentication |
| ICP | Internal Connection Point |
| LDAP | Lightweight Directory Access Protocol |
| OE | operational environment |
| RFC | Request for Comments |
| SAR | Security Assurance Requirement |
| SF | Security Function |
| SFP | Security Function Policy |

[4] The term "TOE account" may be used to mean the human or IT entity that is able to interact with the TOE by means of that account (which requires presentation of a valid password before interaction can begin). Note that the *database owner* user differs from other authorised TOE users in that there is no associated TOE account, but only account(s) maintained by the operating environment.

| Acronym | Meaning |
|---|---|
| SFR | Security Functional Requirement |
| SHA | Secure Hash Algorithm i.e. SHA256 (bits) |
| ST | Security Target |
| TLS | Transport Layer Security |
| TSF | TOE Security Functionality |
| TSFI | TSF Interfaces |
| TSS | TOE Summary Specification |

## 1.4 Revision History

| Issue | Date | Description | Author |
|---|---|---|---|
| 1.5 | 24/07/17 | Update to Issued | RG (Egress) |

# 2 Introduction

## 2.1 ST Reference

12. This ST document is identified as:

Egress Switch Secure Email & File Transfer v4.8 Security Target,
Reference 56133.LFL/T280-cons.ST.1,
Version 1.5 of 24 July 2017.

13. It was prepared by CGI IT UK Ltd on behalf of Egress Software Technologies Limited.

## 2.2 TOE Reference

14. The TOE is identified as:

Egress Switch Secure Email & File Transfer v4.8, supplied by Egress Software Technologies Limited.

## 2.3 TOE Overview

15. The Egress Switch product as a whole consists of various software components, subsets of which can be installed and configured to provide a range of secure services based on the use of cryptography. One such subset forms the TOE[5], which may be classed as a type of product that provides an encrypted email service.

16. The TOE is intended to be installed as part of a secure network, in order to allow users to exchange sensitive emails with users in another secure network (which has another instance of the TOE installed in it).[6]

17. The main security features of the TOE are as follows:

1) The TOE enables users to send and receive encrypted emails (messages and attachments) over insecure communication channels using e.g. SMTP. Symmetric encryption (AES 128/192/256-CBC) is used; it is not necessary for individual users to have public or private keys;

2) The TOE implements *policies* whereby a user can control who can receive the decryption key for an email he has sent, and how such recipients are to be authenticated before the key is released to them. The key is requested and sent using a TLS-protected channel;

3) The TOE provides an audit capability to record details of, for example, requests for decryption keys, *policy* changes, and TOE configuration changes.

[5] Another subset provides a secure file transfer service using e.g. FTP; that product should not be confused with the TOE, where “File Transfer” in the TOE’s reference means the transfer of files as attachments to email messages.

[6] Other deployment scenarios are possible but are outside the scope of this ST. (For example, Egress provides a subscription service whereby an encrypted email can be decrypted by the recipient irrespective of whether he has direct access to an instance of the TOE.)

18. The TOE can operate with a variety of supporting platforms and products. In general terms, such support (i.e. the operating environment) must include or provide:

1) Server and client operating systems;
2) An email server and clients;
3) An SQL database management system;
4) A cryptographic library (the TOE does not directly implement e.g. the AES algorithm);
5) Communication services (for both internal and external communications), including TLS-protected web services.

19. The logical scope of the TOE (and its required deployment scenario and operating environment) is defined in the next section, together with further details of how the TOE functions.

## 2.4 TOE Description

### 2.4.1 Components and their functions

20. The TOE consists of three logical software components, identified as follows:

1) Egress Switch Gateway (ESG), which is server based;
2) Egress Switch Client (ESC), which is client based;
3) Egress Switch Infrastructure (ESI), which is server based.

21. The ESI consists of four sub-components, identified as:

1) External Connection Point (ECP);
2) Internal Connection Point (ICP);
3) Authentication Server (AuS);
4) Database Server (DBS).[7]

22. These components and sub-components are depicted (simplistically) in Figure 1 overleaf, which shows two (or more) instances of the TOE communicating over an insecure network. Figure 1 is followed by a table that outlines the functions of the (sub) components. Note that:

1) It is possible for the TOE to operate with or without the ESC component being installed on (some or all) of the email client machines;
2) The ICP includes a means of configuring and managing the TOE;
3) All *users* and most *administrators* of the TOE need to have their own TOE account (which is quite separate from any OE accounts they also have).[8]

[7] The ECP, ICP and AuS ESI sub-components include TOE software, but the “Database Server” represents TSF data that is ultimately handled by a DBMS in the operational environment. The ECP and the ICP have the same TOE software installed on them, but are configured differently (e.g. many potential facilities are disabled in the ECP). They are the only (sub) components that offer facilities (APIs) accessible to external users. The allocation of the ESG component and the ESI sub-components to physical server platforms within a secure network is flexible, but is not considered further in this ST.

[8] Further details of TOE accounts and administrative functions are given in Section 7.2.

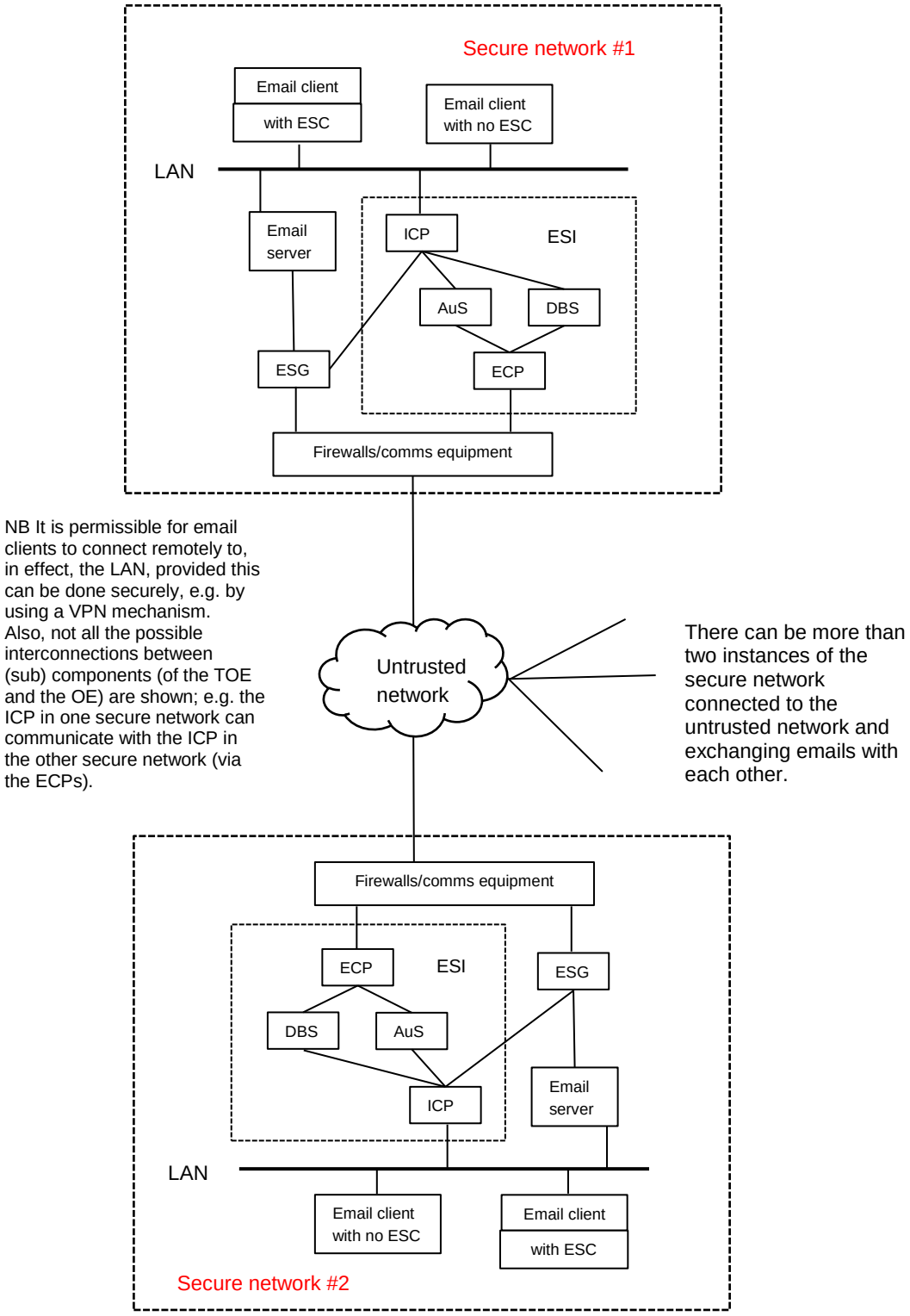


**Figure 1: TOE components, sub-components and deployment**

| Component | Function |
|---|---|

| Component | Function |
|---|---|
| **Egress Switch Gateway (ESG)** | The ESG is located at the boundary of a secure network, and it processes incoming and outgoing emails destined for, or coming from, the network's "vanilla" email server. Processing is dictated by the TOE's *policies* (see Section 1.3), which the ESG obtains from the DBS.<br>If a *corporate policy* dictates that an outgoing email must be encrypted then the ESG encrypts it with a one-time AES key (unless that has already been done by an ESC component) and releases it (as part of a *package* with a unique id) for onward transmission over an insecure network. The ESG also creates a *packreg* object (containing the key and other details of the email/*package*), and stores ("registers") this in the DBS. The *packreg* may be linked to a *onemail policy* (and possibly *corporate policies* as well) which specifies any additional authentication requirements that must be met before the email's decryption key can be released.<br>When the ESG receives an encrypted email (from another instance of the TOE) contained in a *package*, then, for recipient(s) that have ESC available, it can simply pass the email on to them via the "vanilla" email server. For other recipient(s), the ESG attempts to decrypt the email, as explained in Paragraph 23 below.[9]. |
| **Egress Switch Client (ESC)** | The ESC is installed in a *user*'s client machine and enables him to encrypt an outgoing email (before it is sent to the "vanilla" email server) and to decrypt an (encrypted) incoming email retrieved from the "vanilla" email server. Encryption and decryption is done in a similar manner as is done by the ESG (the only difference being that obtaining the decryption key for an email that was sent from within the same secure network does not involve another TOE instance). |
| **External Connection Point (ECP)** | The ECP is located at the boundary of a secure network, and handles incoming (HTTPS) requests from another TOE instance – made on behalf of recipient(s) of an encrypted email - for a decryption key. The ECP authenticates the sender's X.509 certificate (checking that it has not been revoked), then passes the request on to the ICP. If the ICP decides the request is valid it provides a copy of the key, which the ECP then forwards on to the requesting TOE instance.[10]<br>The ECP also forwards (HTTPS) requests for a decryption key to another TOE instance on behalf of the ESG (or an ESC), as explained in Paragraph 23 below. |

[9] If decryption fails for some reason, e.g. the decryption key cannot be retrieved, then the ESG's behaviour proceeds as per the relevant *policy*, which may be, for example, to forward the encrypted email to the recipient(s) with a message saying that decryption failed.

[10] Obviously, if an authentication or confirmation check fails then the key is not released.

| Component | Function |
|---|---|
| **Internal Connection Point (ICP)** | The ICP is located within a secure network, and handles requests for a decryption key as explained in Paragraph 23ff below.<br>It also includes a facility for configuring and managing the TOE (which may be accessed from a browser via a web interface if the ICP host machine is appropriately configured). However, a user attempts to access it, the management facility requires that user to first supply valid TOE account details (username and password). |
| **Authentication Server (AuS)** | The AuS authenticates TOE account usernames and passwords.[11] (For example, when a user attempts to access the TOE management facility, the ICP requests the AuS to authenticate the user-supplied username and password.) |
| **Database Server (DBS)** | The DBS stores TSF and user data, in particular *policies* and *packregs*, and the TOE's audit records. This collection of data is referred to as the TOE database. |

23. If the ESG (in TOE#1 say) needs to decrypt an incoming email then it must obtain a copy of the decryption key from the TOE instance that sent it (TOE#2 say); the process is as follows:

1) The ESG, using its TOE service account, requests the ICP to obtain a copy of the key for the incoming email (identified by its container *package* id) ;

2) The ICP requests the AuS to verify the service account password supplied by the ESG, then (if the password is valid) passes the request on to the TOE#2 ICP (via the ECP sub-components);

3) The TOE#2 ICP checks the applicable TOE#2 *policies* to confirm that a copy of the key can be released to the TOE#1 ESG;[12] if it can then the key copy is sent to the TOE#1 ESG (via the same channels that carried the ESG's request to the TOE#2 ICP);

4) The ESG uses the key to decrypt the email, before passing it on to the "vanilla" email server.

24. If an ESC in TOE#1 needs to decrypt an email that was sent from within TOE#1, then it requests a copy of the key from the ICP, which requests the AuS to verify the TOE account password supplied by the ESC; if the password is valid then the ICP checks the applicable TOE#1 *policies* to confirm that a copy of the key can be released to the ESC.

[11] The AuS interacts with an LDAP server (not shown on Figure 1) on which all TOE account details are stored. This OE server is relied upon to protect those details, in particular the passwords, from unauthorised access.

[12] A *policy* may stipulate that the request is supported by some additional credential(s), which may involve further exchanges between the two ICPs (e.g. if a valid response to a challenge is required). For the purposes of this ST it may be assumed that all necessary credentials are included with the key request.

25. Note that in a typical installation of the TOE (in one secure network) either all or none of the email clients that may receive encrypted email have ESC installed on them. In the deployment shown in Figure 1 it is conceivable that the client with ESC sends (or copies) an encrypted email to the client without ESC; however, in this (unlikely) situation it is possible that the email could be forwarded from the ESC-less client to reach the ESG as if it had been sent by another TOE instance. The ESG could then obtain a copy of the key from the ICP, decrypt the email, and send it via the “vanilla” email server to the ESC-less client.

26. Figure 2 provides a more detailed (than Figure 1) depiction of how the TOE operates. The diagram does not show the four ESI sub-components as separate icons (a single “ESI = Key Server” icon is shown instead).

27. Section 7 below outlines how the TOE’s security functional requirements (which are specified in Section 6.1) are implemented, both in terms of security functions and the (sub) components described above. Section 7 also gives further details of:

1) TOE accounts and administrative functions;

2) Security attributes.

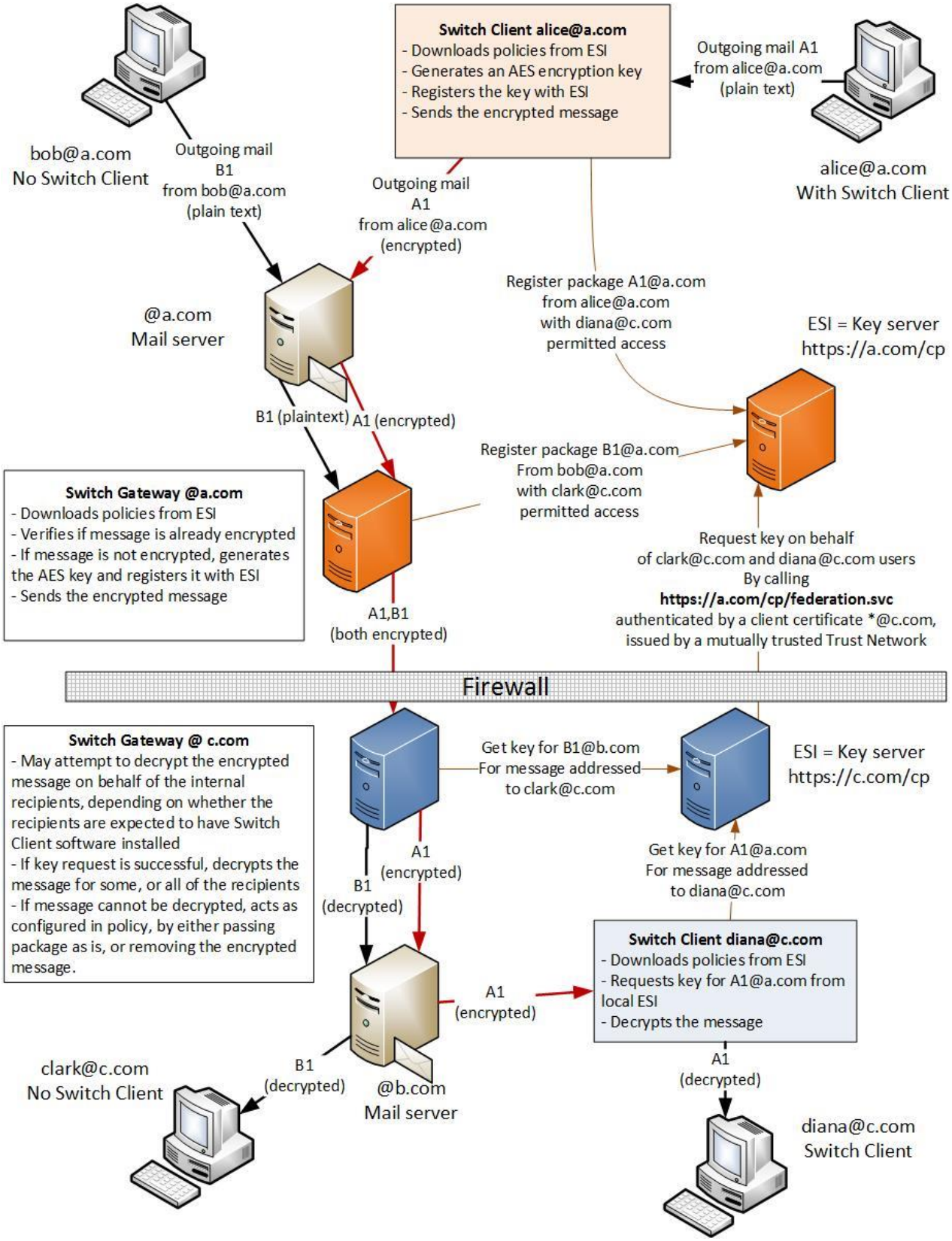


**Figure 2: How the TOE operates**

### 2.4.2 Scope of the TOE

28. Figure 3 below depicts, by shading, the logical scope of the TOE as installed in one secure network. The TOE relies on the unshaded components of the depicted network (part of the TOE's operating environment) to operate correctly.[13]

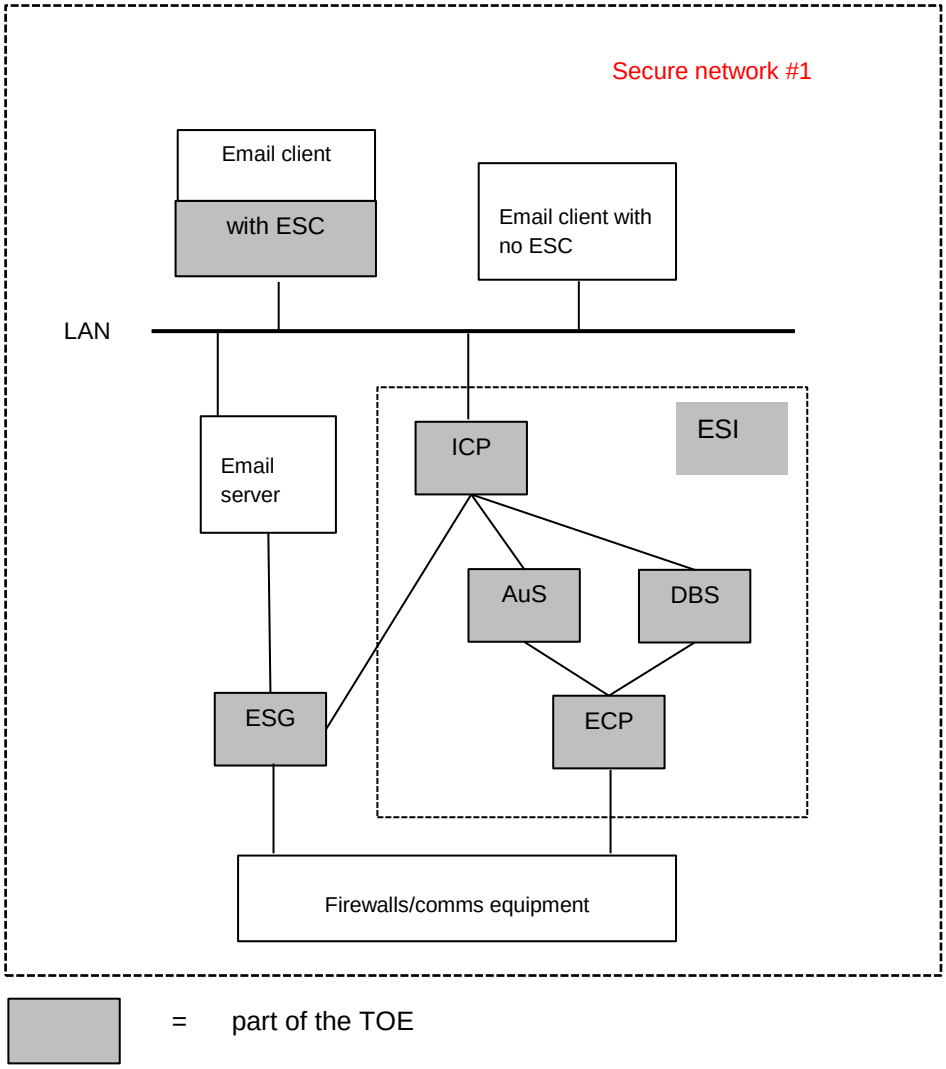


**Figure 3: Logical scope of the TOE**

[13] Not all such components are depicted, e.g. the platform(s) on which the TOE components are installed.

### 2.4.3 Evaluated configuration

29. As previously noted, the TOE is designed to operate in a variety of deployment scenarios, and to be capable of running on a variety of platforms. However, for the purposes of this ST, the TOE is assumed to be deployed as indicated in Figure 1, and to be running in a Microsoft Windows-based operational environment in accordance with the following table.[14]

30. Note that the results of evaluating the TOE against this ST cannot be automatically extrapolated to apply to the TOE installed in a different deployment scenario or operational environment.

| Component | Platform/Product |
|---|---|
| **TOE ESG** | Microsoft Windows Server 2008 R2 (64 bit), or<br>Microsoft Windows Server 2012 R2 (64 bit) or<br>Microsoft Windows Server 2016 (64 bit) |
| **TOE ESC** | Microsoft Windows 7 (32/64 bit), or<br>Microsoft Windows 8.1 (32/64 bit), or<br>Microsoft Windows 10 (32/64 bit)<br>(all including Microsoft Outlook) |
| **TOE ECP** | As TOE ESG |
| **TOE ICP** | As TOE ESG |
| **TOE AuS** | As TOE ESG |
| **TOE DBS** | As TOE ESG (plus Microsoft SQL Server 2008 R2 or 2012) |
| | |
| **OE email server** | As TOE ESG, including Microsoft Exchange Server |
| **OE user PC with no ESC** | As TOE ESC |
| **OE LDAP server** | As TOE ESG (plus Microsoft AD LDS) |

[14] Where the table offers a choice of platform for a component, one platform will be selected to form part of the test system used during the evaluation process.

# 3 Conformance Claims

## 3.1 Common Criteria Conformance

31. This ST is conformant to the Common Criteria for Information Technology Security Evaluation, Version 3.1 Revision 4 Final of September 2012, as follows:

1) Part 2 extended with the components specified in Section 8 below;
2) Part 3 conformant;
3) EAL2 conformant.

## 3.2 Protection Profile Conformance

32. This ST does not claim conformance to any protection profile.

# 4 Security Problem Definition

## 4.1 Threats

### 4.1.1 Assets requiring protection

33. The primary (user) assets to be protected by the TOE and its operational environment (OE) are the (plaintext) email messages and attachments that are processed by the TOE.

34. Secondary assets to be protected by the TOE and its OE are:

1) Audit records generated by the TOE;

2) The TOE's configuration data and software; in particular its *policies* and cryptographic keys;[15]

3) OE software that the TOE relies on (e.g. software providing cryptographic services).

35. Note that:

36. 1) A one-time key generated by the TOE to encrypt an email effectively becomes a primary asset in place of that email until the email is decrypted.

37. 2) The TOE is primarily concerned with protecting the confidentiality and integrity of its assets; no claims are made about the availability of those assets (which may be affected by, for example, the failure of a communication device).

### 4.1.2 Threat agents

38. Potential threat agent types who may accidentally, or deliberately attempt to, compromise the above assets are:

1) *hacker*, i.e. someone who is not authorised to use or administer the TOE or to access or configure any of its assets. A *hacker* is either *internal*, i.e. authorised to use or administer some other part(s) of the OE, or *external* (i.e. not authorised to use or administer any part of the OE - in particular someone who can intercept emails sent over the Internet by a *user* of the TOE);

2) *administrator*, i.e. an authorised administrator of the TOE (and possibly of other parts of the OE). For the purposes of this part of the ST all administrators are considered to have the same level of administrative privilege[16], and they are not considered to be *users* of the TOE[17];

3) *user*, i.e. an authorised user of the TOE who is not an *administrator*.[18]

---

[15] Including random numbers generated during the creation of keys.

[16] The different levels of administrative privilege that the TOE does support are defined in Section 7.2.

[17] An *administrator* may use the TOE, but in general his privileges would enable him to defeat a countermeasure to a threat posed by a *user*, so it is more convenient to treat the two types of threat agent as distinct. Note that an administrator of the DBMS that underpins the TOE's Database Server sub-component is treated as an *administrator*.

[18] Hence, someone who can administer other parts of the OE (but not the TOE) and use the TOE is considered to be a *user*.

### 4.1.3 Statement of threats

39. The threats to be countered by the TOE and its operational environment are listed in the following table. The assumed attack potential of a threat agent is *basic*.[19]

| Threat id | Description |
|---|---|
| **T-Admin** | An *administrator* attempts to subvert the operation of the TOE, e.g. by adding other addresses to an encrypted email's recipient list, amending *policy* data, accessing decryption keys, etc. |
| **T-Hacker-Int** | An *internal hacker* attempts to subvert the operation of the TOE, e.g. by acquiring *administrator* privileges. |
| **T-Hacker-Ext** | An *external hacker* intercepts encrypted emails and related traffic (e.g. the decryption key) in transit between senders and receivers, or attempts to gain access to the OE. |
| **T-User** | A *user* attempts to gain *administrator* privileges, or to access TOE assets in an unauthorised manner (e.g. to access emails sent to or by another *user*). |

## 4.2 Organisational Security Policies

40. No organisational security policies (OSPs) are specified for the TOE or its operational environment.

## 4.3 Assumptions

41. Assumptions made on the operational environment are listed in the following table.
Note also that there is an implicit assumption that a correctly configured OE functions correctly, e.g. emails are delivered to their intended recipients' mailboxes, communication devices deliver packets to the correct machines.

[19] See [CEM], Section B.2, for an explanation of attack potential. A threat agent, particularly an administrator or internal hacker, may well have a higher attack potential than *basic*, but for EAL2 a TOE need only demonstrate resistance to attackers with a *basic* attack potential.

| Assumption id | Description |
|---|---|
| **A-Crypto-Lib** | The OE includes a library of cryptographic algorithms and functions that the TOE can use to support it – the TOE – in meeting its cryptographic requirements. Those algorithms and functions that the TOE does use have been certified elsewhere as meeting the relevant crypto standards that are specified in the TOE's SFRs. |
| **A-Good-Staff** | *Administrators and users*, and administrators and users of the OE, are "good", i.e. they are adequately trained in administering/using the TOE and/or the OE in a secure manner, and they are trusted to follow that training in practice.[20] |
| **A-Logical-Access** | The OE includes facilities for identifying and authenticating a user,[21] then controlling which functions and data provided by or handled within the OE that user can access, and in what manner (e.g. read only); facilities for generating audit records of users' activities; and an email facility for sending emails to, and receiving emails on behalf of, their intended recipients.[22] |
| **A-Physical-Access** | The OE (including the TOE) is secured against unauthorised physical access. |
| **A-Secure-Comms** | The communication channels used by TOE components and sub-components to communicate with each other are secured so that data being carried over those channels is protected from modification or disclosure. |
| **A-Secure-Config** | Hardware and software products in the OE (including the TOE) are received, installed, configured and managed in accordance with their suppliers' instructions;[23] configuration and management is done using the facilities provided by the TOE and assumed by A-Logical-Access. |

[20] Clearly, if this standard assumption was invariably upheld then *administrators*, *users* and *internal hackers* would cease to be threat agents; however, any individual may make accidental errors, and it can never be guaranteed that no-one – even a highly trusted *administrator* - will ever deliberately attempt to attack the TOE's assets.

[21] "User" here means a human or an IT entity attempting to interact with some part(s) of the OE (including the TOE); see [CC] Part 1, Section 4.1.

[22] This email subsystem must be such that the TOE can function in conjunction with it; for example, the ESC functions can be integrated with the send and retrieve functions of the client part of the email subsystem.

[23] These instructions include, in particular, instructions to apply operating system security patches in a timely manner, to deploy up-to-date anti-virus software in the OE, and to regularly inspect audit records for evidence of malpractice.

# 5 Security Objectives

## 5.1 Security Objectives for the TOE

42. The security objectives for the TOE are listed in the following table.

| TOE objective id | Description |
|---|---|
| **O-Admin** | The TOE shall provide facilities for *administrators* to manage the TOE (e.g. to create and maintain TOE accounts and *corporate policies*). See Section 7.2 for more details of *administrators* and TOE accounts. |
| **O-Audit** | The TOE shall generate audit records (of significant events relating to the TOE's assets), and shall be able (where possible) to associate audit records with users. |
| **O-Crypto** | The TOE shall employ AES-128 or 256 cryptographic algorithms and functions to implement its primary function (i.e. to secure the confidentiality and integrity of email messages and attachments transmitted over an insecure network). |
| **O-Key-Release** | The TOE shall control access to, and export of, secret cryptographic keys that it generates when encrypting emails (messages and attachments). Such a key (for a given email) shall be released only to the sender and the intended recipients of that email, and shall be delivered to them using a secure communications channel. |

## 5.2 Security Objectives for the Operational Environment

43. The security objectives for the operational environment are listed in the following table.

| OE objective id | Description |
|---|---|
| **OE-Crypto-Lib** | The OE shall include a library of cryptographic algorithms and functions that the TOE can use to support it – the TOE – in meeting its cryptographic requirements. Those algorithms and functions that the TOE does use shall have been certified elsewhere as meeting the relevant crypto standards that are specified in the TOE's SFRs. |
| **OE-Good-Staff** | The OE shall ensure that Administrators and users of the TOE are "good", where they shall be adequately trained in administering / using the TOE/OE through stringent Egress Switch training programme(s) and/or absorbing supporting preparative & operational guidance documentation to reduce the likelihood of deliberate or inadvertent (whether through incompetence, carelessness or naivety) misuse.[20] |
| **OE-Logical-Access** | The OE shall include facilities for identifying and authenticating a user[21], then controlling which functions and data provided or handled by the OE that user can access, and in what manner (e.g. read only); facilities for generating audit records of users' activities; and an email facility for sending emails to, and receiving emails on behalf of, their intended recipients.[22] |
| **OE-Trust-Network** | The OE shall provide a mutually trusted Trust Network that issues client certificates as shown in Figure 2, above. |

| OE objective id | Description |
|---|---|
| **OE-Physical-Access** | The OE (including the TOE) shall be secured against unauthorised physical access. |
| **OE-Secure-Comms** | The communication channels used by TOE components to communicate with each other shall be secured so that data being carried over those channels is protected from modification or disclosure. |
| **OE-Secure-Config** | Hardware and software products in the OE (including the TOE) shall be received, installed, configured and managed in accordance with their suppliers' instructions[23]; configuration and management shall be done using the facilities provided by O-Admin and specified by OE-Logical-Access. |

## 5.3 Rationale

44. The following table traces objectives to threats and assumptions. The entries in the "Notes" column justify why the objectives counter the threats.

| Threat/Assumption | Objective(s) | Notes |
|---|---|---|
| T-Admin | O-Audit<br><br><br><br><br>OE-Good-Staff<br>OE-Logical-Access<br>OE-Secure-Config | O-Audit mitigates against the threat of an *administrator* misusing his privileges to attack the TOE or its assets, e.g. because details of his actions (including his identity) will be captured in audit record(s).<br><br>The OE objectives diminish the threat. |
| T-Hacker-Int | O-Admin<br><br><br><br>OE-Good-Staff<br>OE-Logical-Access<br>OE-Secure-Comms<br>OE-Secure-Config | O-Admin, supported by the OE objectives, counters the threat of an *internal hacker* gaining unauthorised access to the TOE or its assets.<br>The OE objectives diminish the threat. |

| Threat/Assumption | Objective(s) | Notes |
|---|---|---|
| T-Hacker-Ext | O-Admin | O-Admin diminishes the threat of an external hacker being able to successfully attack the TOE should he manage to gain unauthorised access to the OE (by breaching the OE defences). |
| | O-Crypto | O-Crypto counters the threat of an *external hacker* being able to read or modify the contents of an intercepted email in a meaningful way. |
| | O-Key-Release | O-Key-Release counters the threat of an external hacker successfully requesting the decryption key for an intercepted email. |
| | OE-Trust-Network | OE-Trust-Network supports O-Key-Release within the operational environment in order to issue client certificates. |
| | OE-Crypto-Lib | OE-Crypto-Lib supports O-Crypto. |
| | OE-Physical-Access | OE-Physical-Access directly counters the threat of an *external hacker* attempting to gain unauthorised physical access to the OE. |
| T-User | O-Admin | O-Admin, supported by the OE objectives, counters the threat of a *user* gaining unauthorised access to the TOE's assets or to *administrator* privileges. |
| | OE-Good-Staff<br>OE-Logical-Access<br>OE-Secure-Config | The OE objectives diminish the threat. |
| | | |
| A-Crypto-Lib | OE-Crypto-Lib | This objective directly upholds the assumption. |
| A-Good-Staff | OE-Good-Staff | This objective directly upholds the assumption. |
| A-Logical-Access | OE-Logical-Access | This objective directly upholds the assumption. |
| A-Physical-Access | OE-Physical-Access | This objective directly upholds the assumption. |
| A-Secure-Comms | OE-Secure-Comms | This objective directly upholds the assumption. |
| A-Secure-Config | OE-Secure-Config | This objective directly upholds the assumption. |

# 6 Security Requirements

## 6.1 Security Functional Requirements

### 6.1.1 Introduction

45. The SFRs for the TOE are specified in the following subsections. The components are the extended components FCS_CKM_EXT.1 and FCS_TLS_EXT.1 specified in Section 8, plus others drawn from the CC Part 2 families FAU_GEN and STG; FCS_CKM and COP; FDP_ACC and ACF; FIA_UAU and UID; FMT_MOF, MSA, SMF and SMR; FPT_STM; and FTP_ITC.

46. Words in the functional elements that appear in square brackets are the result of permitted operations on those elements, and any minor editorial changes from the text of [CC] Part 2 are indicated by underlining.

47. Note that an SFR may be satisfied wholly or partly by the TOE correctly using appropriate facilities provided in the operational environment (e.g. APIs for the OE's crypto library), as opposed to implementing the SFR directly (e.g. by including an implementation of the AES algorithm within its own object code, or maintaining a reliable source of time stamps which is distinct from that maintained by the OE). Hence, for example, SFR FPT_STM.1.1 (The TSF shall be able to provide reliable time stamps) could be satisfied by the TSF being able to call an OE API that returned the current time (provided that OE API was judged to have been implemented correctly in the OE).[24]

### 6.1.2 Security audit (FAU)

**FAU_GEN.1** Audit data generation

Dependencies: FPT_STM.1 (Reliable time stamps).

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

a) Start-up and shutdown of the audit functions are logged in both the Windows Event log and the audit logs of the individual components themselves;

b) All auditable events for the [not specified] level of audit; and

c) [The auditable events defined in the following table (overleaf)].

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

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the ST, [other audit relevant information as specified in the following table (overleaf)].

**Application note:** The TOE is also capable of recording details of many other auditable events[25], but they are not specified in this ST. Such as internal audit events (too many to list), including all Windows Event Logs, IIS etc. audit events.

[24] The TOE evaluators will make such judgements, which in general can be based solely on an examination of the CC/FIPS certification status of the relevant OE component and its security target.

[25] Such as those indicated (at various levels of audit) in [CC] Part 2 for the SFR components that are included in this ST.

| Auditable event (type) | Other audit relevant information |
|---|---|
| An attempt to establish a TLS channel between the TOE and another instance of the TOE | The IP addresses of the two ends of the channel; reason for failure (if the attempt failed). |
| Checking the validity of an externally-presented X.509 certificate (including certificate revocation checking) | The certificate id and its CA; reason for rejecting the certificate (if a check failed). |
| An attempt to access a decryption key (contained in a *package*) | The *package* id; reason for denying access (if the attempt failed). |
| An attempt to decrypt an encrypted email (contained in a *package*) | The *package* id; reason why decryption failed (if it did fail). |
| | |
| An attempt to use a *super-user* management function (as defined in Section 7.2 above) | The id of the *organisation* partition or the *corporate policy* that was the target of the partition or *policy* management function; The TOE account id and its attributes (e.g. password) that was the target of the TOE account management function. |
| An attempt to use an *organiser* management function (as defined in Section 7.2 above) | The id of the *corporate policy* that was the target of the *policy* management function; The TOE account id and its attributes (e.g. password) that was the target of the TOE account management function. |
| An attempt to use a *user* management function (as defined in Section 7.2 above) | The id of the *onemail policy* that was the target of the *policy* management function. |

48. **FAU_GEN.2** User identity association
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.

49. **FAU_STG.1** Protected audit trail storage
Dependencies: FAU_GEN.1 (Audit data generation)
**FAU_STG.1.1** The TSF shall protect the stored audit records in the audit trail from unauthorised deletion.
**FAU_STG.1.2** The TSF shall be able to [detect] unauthorised modifications to the stored audit records in the audit trail.

### 6.1.3 Cryptographic support (FCS)

50. **FCS_CKM.1(1)** Cryptographic key generation (for TLS session key agreement)
Dependencies: FCS_COP.1 (Cryptographic operation)
FCS_CKM.4 (Cryptographic key destruction)
FCS_CKM_EXT.1 (Cryptographic key storage)
FCS_RBG_EXT.1 (Random bit generation)
**FCS_CKM.1.1(1)** The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm: [ECDH using NIST P-256 curve] and specified cryptographic key size [256 bits] that meet the following: [NIST SP 800-56A].

51. **FCS_CKM.1(2)** Cryptographic key generation (symmetric keys)
Dependencies: FCS_COP.1 (Cryptographic operation)
FCS_CKM.4 (Cryptographic key destruction)
FCS_CKM_EXT.1 (Cryptographic key storage)
FCS_RBG_EXT.1 (Random bit generation)
**FCS_CKM.1.1(2)** The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [generate a random number] and specified cryptographic key sizes [256 bits] that meet the following: [as specified in FCS_RBG_EXT.1].

52. **FCS_CKM.4** Cryptographic key destruction
Dependencies: FCS_CKM.1 (Cryptographic key generation)
**FCS_CKM.4.1** The TSF shall destroy cryptographic keys by overwriting the key with binary zeros once.

53. **FCS_CKM_EXT.1** Cryptographic key storage
Dependencies: FCS_CKM.1 (Cryptographic key generation)
**FCS_CKM_EXT.1.1** The TSF shall store [symmetric email decryption] cryptographic keys in accordance with a specified cryptographic key storage method [AES Key Wrap] that meets the following: [RFC 3394 and/or NIST SP 800-38F].

54. **FCS_COP.1(1)** Cryptographic operation (symmetric encryption)
Dependencies: FCS_CKM.1 (Cryptographic key generation)
**FCS_COP.1.1(1)** The TSF shall perform [symmetric encryption and decryption] in accordance with a specified cryptographic algorithm [AES operating in CBC mode] and cryptographic key sizes of [128 or 192 or 256 bits] that meet the following: [FIPS Pub 197, NIST SP 800-38A and -38D].

55. **FCS_COP.1 (2)** Cryptographic operation (hashing)
Dependencies: FCS_CKM.1 (Cryptographic key generation)
**FCS_COP.1.1 (2)** The TSF shall perform [cryptographic hashing] in accordance with a specified cryptographic algorithm [SHA-256] and message digest size [256 bits] that meet the following: [FIPS Pub 180-3 or 180-4].

56. **FCS_COP.1(3)** Cryptographic operation (digital signature)
Dependencies: FCS_CKM.1 (Cryptographic key generation)
**FCS_COP.1.1(3)** The TSF shall perform [digital signature operations] in accordance with a specified cryptographic algorithm:
a) [DSA] and cryptographic key sizes of [at least 1536/192 bits] that meet the following: [FIPS Pub 186-2], or
b) [ECDSA using NIST P-256 curve] and cryptographic key size [256 bits] that meet the following:
[ANSI X9.62 or FIPS Pub 186-4].

57. **FCS_COP.1(4)** Cryptographic operation (HMAC)
Dependencies: FCS_CKM.1 (Cryptographic key generation)

58. HMAC is used for non-primary functions of Egress Switch, such as ensuring the integrity of tokens issued by the Connection Point service to the Administration web interface.

59. The Egress Switch Package does not use HMAC.

60. **FCS_COP.1.1(4)** The TSF shall perform [keyed-hash message authentication] in accordance with a specified cryptographic algorithm [HMAC-SHA-256] and cryptographic key and message digest sizes [256 bits] that meet the following: [FIPS Pub 198-1, and FIPS Pub 180-3 or 180-4].

61. **FCS_RBG_EXT.1** Random bit generation
Dependencies: No dependencies
**FCS_RBG_EXT.1.1** The TSF shall implement a deterministic random bit generator in accordance with [NIST SP 800-90A].
**FCS_RBG_EXT.1.2** The deterministic random bit generator shall be seeded by an entropy source that accumulates entropy from [a combination of software-based and (if available) hardware-based noise sources] with a minimum of [256 bits] of entropy which is at least equal to the greatest security strength required for the keys and hashes that the TSF will generate.

62. **FCS_TLS_EXT.1** TLS protocol
Dependencies: FCS_CKM.1 (Cryptographic key generation)
FCS_COP.1 (Cryptographic operation).
**FCS_TLS_EXT.1.1** The TSF shall implement the TLS 1.2 protocol in accordance with RFC 5246 using only the following ciphersuites: TSF relies on Microsoft Windows' built-in Secure Channel TLS implementation. TSF applies Secure Channel configuration to enforce TLS 1.2 with the following ciphersuites:
ECDHE_ECDSA_WITH_AES_128_CBC_SHA_256_P256
ECDHE_ECDSA_WITH_AES_128_CBC_SHA_256_P384
ECDHE_ECDSA_WITH_AES_128_CBC_SHA_256_P512
ECDHE_ECDSA_WITH_AES_128_GCM_SHA_256_P256
ECDHE_ECDSA_WITH_AES_128_GCM_SHA_256_P384
ECDHE_ECDSA_WITH_AES_128_GCM_SHA_256_P512
ECDHE_ECDSA_WITH_AES_256_GCM_SHA_384_P384
ECDHE_ECDSA_WITH_AES_256_GCM_SHA_384_P512
ECDHE_ECDSA_WITH_AES_256_CBC_SHA_384_P384
ECDHE_ECDSA_WITH_AES_256_CBC_SHA_384_P512
ECDHE_RSA_WITH_AES_128_CBC_SHA_256_P256
ECDHE_RSA_WITH_AES_128_CBC_SHA_256_P384
ECDHE_RSA_WITH_AES_128_CBC_SHA_256_P512
ECDHE_RSA_WITH_AES_256_CBC_SHA_384_P256
ECDHE_RSA_WITH_AES_256_CBC_SHA_384_P384
ECDHE_RSA_WITH_AES_256_CBC_SHA_384_P512

These can be further restricted to allow only Suite-B algorithms or only enable particular ciphersuites from the list.

63. **FCS_TLS_EXT.1.2** The TSF shall validate any X.509 certificate presented to it in the course of establishing a TLS channel with an external entity in accordance with the following rules: [certificate/certificate path validation and certificate revocation checking[26] are to be done in accordance with RFC 5280 or RFC 5759].

### 6.1.4 User data protection (FDP)

[26] The OE is relied upon to provide a means by which the TOE can interrogate a suitable CRL server.

64. **FDP_ACC.1** Subset access control
Dependencies: FDP_ACF.1 (Security attribute based access control)
**FDP_ACC.1.1** The TSF shall enforce the [Email Key Access SFP] on the following types of subject, object and operations between them: [a subject acting on behalf of a TOE account; a *package* (object); access to the object by the subject]**.**
Application note: the "TOE account" may be "external", i.e. be a TOE account in another instance of the TOE.
The OE is relied upon to prevent, as far as is possible, direct access to the content of packages by non-TOE accounts.[27]

[27] Such as the *database owner*, or another administrator of the DBMS that underpins the DBS. These roles generally cannot be entirely prevented from directly accessing data held in the TOE database.

65. **FDP_ACF.1** Security attribute based access control
Dependencies: FDP_ACC.1 (Subset access control)
FMT_MSA.3 (Static attribute initialisation)

**FDP_ACF.1.1** The TSF shall enforce the [Email Key Access SFP] to objects based on the following SFP-relevant security attributes:
[a) subject attributes: the TOE account (id) on whose behalf a subject is seeking to access a *package* and associated authentication credentials;
b) object attributes: the TOE accounts and their authentication requirements that may be referenced in the *policies* that are applicable to that *package*].

Application note: The "associated authentication requirements" are explained in Paragraph 66 below.
The "TOE accounts that may be referenced" are the sender of the email contained in the *package* and all addressed recipients of that email.
The "applicable" *policies* include at least the *onemail* policy created when the *package* was created; one or more *corporate policies* may also be applicable, as explained in Paragraph 66 below.
See also Section 7.3 for further explanation of the TOE's security attributes.

**FDP_ACF.1.2** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:
[Access rule: For each *policy* applicable to the *package* in question determine if it is applicable to the TOE account in question; if it is then locate the authorisation requirements (if any) and for each such requirement check that it is satisfied by a credential of the subject in question. If all authorisation requirements in all applicable policies are satisfied then permit the subject to access the package,[28] otherwise deny access].
**FDP_ACF.1.3** The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [none].
**FDP_ACF.1.4** The TSF shall explicitly deny access of subjects to objects based on the following additional rules:
[If the *onemail policy* applicable to the *package* in question is not applicable to the TOE account in question then access is denied].

Application note: When an encrypted email is sent its *onemail policy* includes the ids of the sender and all addressed recipients, i.e. it is applicable to all those TOE accounts. However, the sender may subsequently remove one or more of the recipient ids (e.g. if he realises that he has addressed the email incorrectly).

[28] e.g. to obtain a copy of the decryption key linked to the *package*.

66. In general, the only authentication requirement in an applicable *policy* is that a TOE account id has been authenticated (by password), either by the TOE itself or by another instance of the TOE[29]. However, it is possible for a *corporate* or a *onemail* policy to specify additional requirements, e.g. that the access request has originated from a specific IP address (range).[30]

### 6.1.5 Identification and authentication (FIA)

67. **FIA_UAU.2** User authentication before any action
Hierarchical to: FIA_UAU.1 Timing of authentication
Dependencies: FIA_UID.1 (Timing of identification).
**FIA_UAU.2.1** The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions (apart from the FIA_UID.2 action) on behalf of that user.

Application note: see FCS_TLS_EXT.1 for additional requirements regarding the use of X.509 certificates for I&A purposes; FDP_ACF.1.1 regarding subsequent I&A requirements that may have to be satisfied; and Section 7.2 (Paragraph 89) for a summary of TOE account management facilities (the I&A aspects of which are not specified in any further SFRs in this ST).

68. **FIA_UID.2** User identification before any action
Hierarchical to: FIA_UID.1 Timing of identification
Dependencies: No dependencies.
**FIA_UID.2.1** The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

### 6.1.6 Security management (FMT)

69. **FMT_MOF.1** Management of security functions behaviour
Dependencies: FMT_SMR.1 (Security roles),
FMT_SMF.1 (Specification of management functions).
**FMT_MOF.1.1** The TSF shall restrict the ability to [disable, enable, and determine or modify the behaviour of] the functions
[a) *super-user* functions; b) *organiser* functions; c) *user* functions] to
[a) the *super-user* role; b) the *super-user* role and the *organiser* role; c) the *user* role].

Application note: these functions are defined in Section 7.2, Paragraph 90ff. The SFR's restriction implicitly extends to "use of" the specified functions. For any given function it may not necessarily be possible to e.g. disable it.

[29] In the latter case the credential is passed to the TOE by the other TOE instance, the identity of which is authenticated as part of the FTP_ITC.1 SFR implementation.

[30] In the case of a *corporate policy* such a requirement will normally be somewhat general, e.g. "any key access request from an id ending @a.com must have originated from IP address w.x.y.z".

70. **FMT_MSA.1** Management of security attributes
Dependencies: FDP_ACC.1 (Subset access control)
FMT_SMR.1 (Security roles)
FMT_SMF.1 (Specification of management functions).
**FMT_MSA.1.1** The TSF shall enforce the [Email Key Access SFP] to restrict the ability to [modify] the security attributes [listed in the Section 7.3 tables as being possible to modify] to [the relevant roles listed in the Section 7.3 tables].

Application note: see Section 7.3 for an explanation of the TOE's security attributes.

71. **FMT_MSA.3** (Static attribute initialisation)
Dependencies: FMT_MSA.1 (Management of security attributes)
FMT_SMR.1 (Security roles).
**FMT_MSA.3.1** The TSF shall enforce the [Email key access SFP] to provide [restrictive] default values for security attributes that are used to enforce the SFP.
**FMT_MSA.3.2** The TSF shall allow the [none, i.e. no identified role] to specify alternative initial values to override the default values when an object or information is created.

72. **FMT_SMF.1** (Specification of management functions)
Dependencies: No dependencies.
**FMT_SMF.1.1** The TSF shall be capable of performing the following management functions: [*super-user* functions, *organiser* functions, *user* functions].

Application note: these functions are defined in Section 7.2, Paragraph 90ff.

73. **FMT_SMR.1** (Security roles)
Dependencies: FIA_UID.1 (Timing of identification).
FMT_SMR.1.1 The TSF shall maintain the roles [*super-user*, *organiser*, *and user*].
FMT_SMR.1.2 The TSF shall be able to associate users with roles.

Application note: as previously stated, an administrator of the DBMS that underpins the TOE's Database Server sub-component is treated as an *administrator*. This role (the TOE's *database owner*) is controlled by the OE, but it must have the capability to manage the TOE's database and to create TOE *super-user* accounts.

### 6.1.7 Protection of the TSF (FPT)

74. **FPT_STM.1** (Reliable time stamps)
Dependencies: No dependencies.
**FPT_STM.1.1** The TSF shall be able to provide reliable time stamps.

### 6.1.8 Trusted path/channels (FTP)

75. **FTP_ITC.1** (Inter-TSF trusted channel)
Dependencies: FCS_TLS_EXT.1 (TLS protocol).
**FTP_ITC.1.1** The TSF shall provide a communication channel between itself and another trusted IT product[31] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.
**FTP_ITC.1.2** The TSF shall permit [the local TSF or another instance of the TOE] to initiate communication via the trusted channel.
**FTP_ITC.1.3** The TSF shall initiate communication via the trusted channel for [requesting an email decryption key].

Application note: the trusted channel mechanism should be HTTPS, using TLS as specified in FCS_TLS_EXT.1.

### 6.1.9 SFR dependencies, management functions and auditable actions

76. It can be seen by inspection of the above SFRs that their dependencies (direct and indirect) are satisfied.

77. It can also be seen (by inspection of the above SFRs, Sections 7.2-7.3 and [CC] Part 2) that:

1) Appropriate management functions are specified in Section 6.1.6;
2) Appropriate auditable actions (or events) are specified in Section 6.1.2.

## 6.2 Security Functional Requirements Rationale

78. The following table traces the above SFRs to objectives for the TOE. By inspection of the table, it can be seen that:

1) Each SFR specified above traces to at least one objective for the TOE;
2) Each objective for the TOE identified earlier has at least one SFR tracing to it;
3) All objectives for the TOE will be achieved (to the level of assurance specified by the following SARs) if all the SFRs are satisfied (to the specified assurance level).

[31] Here, “another trusted IT product” is another instance of the TOE, as depicted in Figure 1. The mechanism for meeting FPT_ITC.1 is provided by FCS_TLS_EXT.1.

| Objective | SFR(s) | Notes |
|---|---|---|
| O-Admin | FMT_MOF.1, SMF.1, SMR.1<br><br>FIA_UAU.2, UID.2 | The FMT_* SFRs directly implement the objective.<br>The FIA_* SFRs ensure that *administrators* can be identified and authenticated. |
| O-Audit | FAU_GEN.1, GEN.2, STG.1<br><br>FIA_UID.2<br>FPT_STM.1 | The FAU_* SFRs directly implement the objective.<br>FIA_UID.2 supports FAU_GEN.2.<br>FPT_STM.1 ensures that a reliable time stamp can be included in audit records. |
| O-Crypto | FCS_CKM.1(1)-(2), CKM.4, CKM_EXT.1, COP.1(1)-(4), RBG_EXT.1 | These SFRs directly implement the objective, and also support FCS_TLS_EXT.1. |
| O-Key-Release | FDP_ACC.1, ACF.1<br>FTP_ITC.1<br>FIA_UAU.2, UID.2<br>FMT_MSA.1, MSA.3<br>FCS_CKM_EXT.1<br>FCS_TLS_EXT.1 | The FDP_* and FTP_ITC.1 SFRs directly implement the objective.[32]<br>The FIA_*, FMT_* and FCS_CKM_EXT.1 SFRs support the FDP_* SFRs.<br>FCS_TLS_EXT.1 supports FTP_ITC.1. |

## 6.3 Security Assurance Requirements

79. The SARs for the TOE are those defined by the EAL2 package that is specified in [CC] Part 3.

80. The assurance components that are relevant to the TOE itself are listed in the following table, together with a summary of the “developer actions”, i.e. a summary of what the TOE’s developer (Egress Software Technologies Ltd) has to provide to the evaluators.

81. In addition, the ST (this document) is to be evaluated against the “ASE_” components for EAL2 as listed in [CC], Part 3, and Table 3.

82. Further details of the assurance components are given in [CC] Part 3. The “evaluator actions” for each component are elaborated in [CEM], which details, for example, how the evaluators should process what the developer provides to them.

[32] OE-Secure-Comms means that FPT_ITT (internal TOE TSF data transfer) need not be specified as a TOE SFR.

| [CC] assurance component | Developer to provide | Notes |
|---|---|---|
| **Development** | | |
| ADV_ARC.1 | Security architecture description of the TSF (TOE Security Functionality) | The TOE design has to prevent the TSF being bypassed or tampered with by untrusted active entities. (In practice the ARC description will probably be a separate section or annex in the design description - see ADV_TDS.1.) |
| ADV_FSP.2 | Security-enforcing functional specification of the TSF | Includes a tracing to the SFRs specified in the ST. |
| ADV_TDS.1 | Basic design of the TOE (describe the design in terms of subsystems) | Includes a mapping from the TSFIs (TSF Interfaces, which are described in the functional specification, see ADV_FSP.2) to the subsystems. |
| **Guidance Documents** | | |
| AGD_OPE.1 | Operational user guidance | Describes how to use the TOE in a secure manner, both for normal (unprivileged) *users* and for *administrators* (who are privileged to configure the TOE's security functions). |
| AGD_PRE.1 | The TOE (i.e. a copy of the TOE software), including its preparative procedures[33] | Procedures describe how to accept (i.e. receive), install and set-up the TOE in a secure manner. (Evaluators can apply these procedures to the TOE at the same time as penetration testing - see below, AVA_VAN.2.) |
| **Life Cycle Support** | | |
| ALC_CMC.2 | A reference for the TOE (as provided for AGD_PRE.1);<br>CM (configuration management) documentation and evidence that a CM system is being used | The CM system needs to uniquely identify all configuration items that constitute the TOE. |
| ALC_CMS.2 | Configuration list for the TOE | List should include the items that constitute the TOE. |
| ALC_DEL.1 | Delivery procedures | Include evidence that the developer follows the documented procedures. |
| **Security Target Evaluation** | | |
| ASE_CCL.1 | Conformance claims | |

[33] A suitable (test) operating environment is also to be provided by the developer.

| [CC] assurance component | Developer to provide | Notes |
|---|---|---|
| ASE_ECD.1 | Extended components definition | |
| ASE_INT.1 | ST (introduction) | |
| ASE_OBJ.2 | Security objectives for the operational environment | |
| ASE_REQ.2 | Stated security requirements | |
| ASE_SPD.1 | Security problem definitions | |
| ASE_TSS.1 | TOE summary specification | |
| **Tests** | | |
| ATE_COV.1 | Evidence of test coverage | Evidence to show that the developer's tests cover some of the TSFIs (which are described in the functional specification, see ADV_FSP.2). |
| ATE_FUN.1 | Functional testing (of the TSF) | Developer's test results (test plans and specifications, expected and actual results). |
| ATE_IND.2 | The TOE (as provided for AGD_PRE.1) for independent testing, plus technical support and resources during the testing that are equivalent to those used in the developer's functional testing of the TSF | Independent testing - sample, i.e. evaluators repeat a sample of the developer's tests; the evaluators also conduct their own additional functional tests. This can be done at the same time as penetration testing - see AVA_VAN.2. |
| **Vulnerability Assessment** | | |
| AVA_VAN.2 | The TOE (as provided for AGD_PRE.1) for penetration testing, plus technical support and resources during the testing (see ATE_IND.2) | Evaluators carry out a vulnerability analysis (based on the ADV and AGD evidence), then undertake penetration testing of the TOE. |

## 6.4 Security Assurance Requirements Rationale

83. The CC characterises an EAL2-assured IT product as one that has been "structurally tested".

84. EAL2 is generally considered to be an appropriate target assurance level for IT products that are intended to safeguard the confidentiality and integrity of government or commercial information (including personal data) that is sensitive, but not highly sensitive.[34]

[34] Within the Common Criteria Recognition Arrangement (CCRA) only evaluations up to EAL 2 are mutually recognized (Including augmentation with flaw remediation).

## 6.5 Conclusion

85. The preceding rationales in this ST demonstrate that, if all the security requirements are satisfied, and all security objectives for the operational environment are achieved, then there exists assurance (to the EAL2 level) that the TOE solves the security problem defined in Section 4.

# 7 TOE Summary Specification

## 7.1 Introduction

86. This TSS section:

1) Provides the details of TOE accounts and administrative functions, and of security attributes, that are referenced from the FDP_* and FMT_* SFRs specified in Section 6.1;

2) Outlines the high-level security functions (SFs) that collectively implement all the SFRs specified in Section 6.1;

3) Maps each SFR to the SF(s) and the (sub-) components (see Section 2.4) that implement it.

## 7.2 TOE Accounts and Administrative Functions

87. All *users* and most *administrators* of the TOE need to have their own TOE account in order to access TOE facilities. Use of a TOE account is controlled and audited by the TOE; each account includes a username and password (which are quite separate from Windows usernames and passwords) for identification and authentication purposes.

88. There are three levels of *administrator* account; these are:

1) *database owner*, this is a Windows (not TOE) account that has administrative rights on OE platforms and the DMBS that maintains the TOE's database (held on the DBS). The *database owner* installs the TOE; part of this installation is to create one or more *super-user* TOE accounts. Use of the *database owner* account is controlled and audited by the OE; after the TOE is installed the account should not need to be involved in the day-to-day management of the TOE (although it can be used to create further *super-user* accounts, and to configure the TOE's audit capability);

2) *super-user* is a TOE account that can use the TOE's management facility to, for example, partition the operation of the TOE into *organisations*, create, modify and delete *corporate policies* applicable to *organisations*, and create, modify and delete *organiser* and *user* accounts;[35]

3) *organiser* is a TOE account that can use the TOE's management facility to administer a specific TOE *organisation*, e.g. to create, modify and delete *corporate policies* applicable solely to that *organisation*, and create, modify and delete *user* accounts relating solely to that *organisation*.

89. A *user* is a TOE account that can use the TOE's ESC component; it can also use the TOE's management facility (to a very limited extent) in connection with itself, e.g. to change the password for that account, and change the *onemail policy* relating to an encrypted email it has sent (e.g. to prevent release of the decryption key to a recipient of that email).

[35] Additional restrictions may be applied to a *super-user* account when it is created by the *database owner*, such as permitting use only from specific IP addresses, or requiring a valid Windows account password to be supplied as well as the *super-user* TOE account password.

90. There is one other type of TOE account – a TOE service account - that TOE software running on one (sub-)component may need to use in order to communicate with (or "logon to") another TOE (sub-component). For example, when the ESG first communicates with the ICP the ESG must supply its (the ESG's) account password (which is created and maintained by a *super-user*).[36]

91. In addition to the above TOE account creation and maintenance facilities, the TOE provides a number of other "account management" facilities[37] for administering TOE accounts, including:

1) A password generation facility (although passwords for TOE accounts used by humans are generally created manually);
2) A password expiry facility;
3) A lockout facility (to counter brute force password attacks);
4) Enforcement of restrictions on passwords, e.g. minimum length;
5) Displaying the date and time of the last logon to a TOE account.

92. For the purposes of SFR specification (see Section 6.1.6), the *super-user* management functions (i.e. the administrative facilities available to a *super-user* TOE account) are:

1) A facility to partition the operation of the TOE into *organisations*;
2) Facilities to create, modify and delete *corporate policies* applicable to *organisations*;
3) TOE account management facilities (as outlined above).

93. For the purposes of SFR specification (see Section 6.1.6), the *organiser* management functions (i.e. the administrative facilities available to a given *organiser* TOE account) are:

1) Facilities to create, modify and delete *corporate policies* applicable solely to the *organiser*'s *organisation*;
2) TOE account management facilities (as outlined above).

94. For the purposes of SFR specification (see Section 6.1.6), the *user* management functions (i.e. the administrative facilities available to a given *user* TOE account) are:

1) The facility to change that TOE account's password;
2) The facility to change a *onemail policy* relating to an encrypted email that was sent by that *user* TOE account.

[36] Currently, this Gateway service account is the only TOE service account.

[37] Apart from in Section 6.1.5, account management facilities are not specified in any further detail in this ST.

## 7.3 Security Attributes

95. For the purposes of SFR specification (see Sections 6.1.4 and 6.1.6), the security attributes that underpin how the TOE controls access to email decryption keys are listed in the following tables, together with details of their default values and permitted management operations. This access control is identified as the "Email Key Access SFP"; it is the sole SFP specified in this ST, and it is necessarily specified at a high level of abstraction from the details of how the TOE implements the policy.

96. Note that:

1) A subject[38] is a process, acting on behalf of a TOE account, requesting access to a *package* (which here can be considered to hold the required email decryption key). In order to obtain access the process needs to present one or more "authentication credentials" of the account. For the present purposes each credential may be referred to as a (previously obtained) "token"; there can be various types of token, but there must always be a "password token", i.e. a token which confirms that the claimed account id has been authenticated by the presentation of the correct TOE account password.[39] Another token type might be an "IP token" that confirms the access request originated from IP address w.x.y.z;

2) An object is a *package* that is linked to one or more *policies*. One such policy is the *onemail policy* that relates to the encrypted email in the *package*, and which contains a list of the TOE accounts (ids) who (potentially) can be granted access to the package. The policy also contains, for each such account, a list of the authentication requirements that must be satisfied before access can be granted to that account. One or more *corporate policies* may also be linked to the package (i.e. may be applicable to whether access will be granted or not); each such *policy* here functions in the same way as the *onemail policy*;

3) For convenience, in the second table below (object attributes), relevant information contained in a *policy* is treated as an attribute of the *package* (i.e. is treated as an object attribute).

| Subject attribute | Default value | Management operation(s) |
|---|---|---|
| TOE account id | Set by OE operating system | None |
| Authentication credential(s) | Set by previous TOE software activity, e.g. authenticating the TOE account id at logon time | None |

[38] "Subject" and "object" are as defined in [CC] Part1.

[39] In the case of a request originating from another TOE instance, the confirmation will have been sent as part of the TLS-protected access request communication.

| Object (*package*) attribute | Default value | Management operation(s) |
|---|---|---|
| Id of *onemail policy* | Linked to the *package* by TOE software when the *package* is created | None |
| TOE account ids (listed in the above *onemail policy*) | The sender and recipients of the encrypted email in the *package* | The *user* who sent the email can modify the list of recipient account ids |
| Authentication requirement(s) (listed in the above *onemail policy*) for each of the above TOE accounts | The TOE account id must have been authenticated (by means of the account password)[40] | The *user* who sent the email can add further authentication requirements for some or all of the email recipients |
| | | |
| None, one or more ids of *corporate policies* | Linked to the *package* by TOE software when the *package* is created | None |
| TOE account id(s) (a list constructed from the contents of the above *corporate policies*) | A subset of the sender and recipients of the encrypted email contained in the *package*. (The subset – which may be empty – consists of any recipients of the encrypted email in the *package* that are affected by the contents of the *corporate policies*) | A *corporate policy* can be modified by a *super-user* or (possibly) by an *organiser*. (Note that any *corporate policy* changes made after a *package* has been created will not modify the *package* attributes) |
| Authentication requirement(s) (a list constructed from the contents of the above *corporate policies*) for each of the above TOE accounts | The TOE account ids must have been authenticated (by means of the account password) | See the row above |

## 7.4 Security Functions

### 7.4.1 Summary

97. The TOE (supported by the OE) provides the following security functions (SFs):

1) Audit (including reliable time stamps);
2) Cryptographic Protection;
3) Access Control;
4) Identification and Authentication (I&A);
5) Security Management;
6) Trusted Channel.

98. Each of these SFs is briefly described in the following subsections.

### 7.4.2 Audit (including reliable time stamps)

[40] Where a recipient and the sender are using different instances of the TOE (as is normally the case), say TOE#2 and TOE#1 respectively, then TOE#2 authenticates the recipient and includes confirmation of this in the key request it sends to TOE#1 (see Figure 2 text “Request key on behalf of ….”).

99. Each TOE (sub) component generates its own audit records as required (independently of any audit records generated by the OE) and stores them in the TOE database. The DBS is configured to prevent/detect unauthorised deletion/modification of the audit records.

100. The I&A SF provides user identity information, and the OE provides a reliable time stamp, that form part of each audit record.

101. The Audit SF implements the FAU_GEN.1, GEN.2, STG.1 and FPT_STM.1 SFRs.

### 7.4.3 Cryptographic Protection

102. The Cryptographic Protection SF encompasses all the crypto-related functions/services provided by the TOE. Apart from key destruction (FCS_CKM.4), the TOE does not directly implement any of the SFRs, but implements them indirectly by making use of the crypto facilities provided by the OE (i.e. the FIPS-certified Windows crypto library).

103. The Cryptographic Protection SF implements the FCS_* SFRs.

### 7.4.4 Access Control

104. The Access Control SF ensures that email decryption keys are released only to authorised recipients, as described in Section 7.3 above.

105. The Access Control SF implements the FDP_ACC.1 and ACF.1 SFRs.

### 7.4.5 Identification and Authentication (I&A)

106. I&A of TOE *users* and *administrators* is done by means of usernames and passwords, as described in Section 7.2 above.

107. The Identification and Authentication SF implements the FIA_UAU.2 and UID.2 SFRs.

### 7.4.6 Security Management

108. The Security Management SF ensures that the administration of TOE accounts and security attributes can be done by authorised *administrators* only, as described in Section 7.2 above.

109. The Security Management SF implements the FMT_MOF.1, MSA.1, MSA.3, SMF.1 and SMR.1 SFRs.

### 7.4.7 Trusted Channel

110. The Trusted Channel SF uses that part of the Cryptographic Protection SF that implements FCS_TLS_EXT.1 (i.e. establishes a TLS 1.2 channel) to provides a secure inter-TSF communications channel.

111. The Trusted Channel SF implements the FTP_ITC.1 SFR.

## 7.5 Implementation of SFRs

112. The following table indicates how - in terms of the SFs outlined in Section 7.4, and the (sub) components and deployment scenario introduced in Section 2.4 - the TOE, supported by the OE, implements each of the SFRs specified in Section 6.1.

| SFR | SF(s) | (Sub-)component(s) | Notes |
|---|---|---|---|
| FAU_GEN.1 | Audit | ECP, ICP, ESG, ESC | |
| FAU_GEN.2 | Audit | ECP, ICP, ESG, ESC (and OE) | |
| FAU_STG.1 | Audit | DBS | |
| | | | |
| FCS_CKM.1(1)-(2) | Cryptographic Protection | ESG, ESC, ECP (and OE) | The TOE uses and relies on the OE's library of cryptographic algorithms and functions. |
| FCS_CKM.4 | Cryptographic Protection | ESG, ESC, ECP, DBS | A persistent email decryption key is deleted from the DBS when the TOE account that created the key is deleted by an *administrator*, and the *administrator* explicitly chooses to delete the key (rather than assign it to another existing TOE account).<br>All persistent keys stored in the DBS can be destroyed by the *database owner* as part of the TOE uninstallation process.<br>Temporary keys, such as TLS session encryption keys, are stored in memory only and are destroyed as soon as they are no longer needed. |
| FCS_CKM_EXT.1 | Cryptographic Protection | ESG, ESC, ECP, DBS | Email decryption keys (held within *packreg* objects in the DBS) are themselves encrypted (by so-called "server" keys). The server keys are held in a file, stored in the DBS, accessible only by *administrators*. |
| FCS_COP.1(1)-(4) | Cryptographic Protection | ESG, ESC, ECP (and OE) | The TOE uses and relies on the OE's library of cryptographic algorithms and functions. |
| FCS_RBG_EXT.1 | Cryptographic Protection | (OE) | The TOE uses and relies on the OE's ability to generate random bits/numbers. |
| FCS_TLS_EXT.1 | Cryptographic Protection | ECP (and OE) | The TOE uses and relies on the OE's implementation of the TLS 1.2 protocol, and its ability to validate X.509 certificates. |
| | | | |
| FDP_ACC.1 | Access Control | ICP, ESG, ESC | |
| FDP_ACF.1 | Access Control | ICP, ESG, ESC | |
| | | | |
| FIA_UAU.2 | Identification & Authentication | AuS | |
| FIA_UID.2 | Identification & Authentication | AuS | |
| | | | |

| SFR | SF(s) | (Sub-)component(s) | Notes |
|---|---|---|---|
| FMT_MOF.1 | Security Management | ICP, AuS | The TOE is installed and initially configured by a *database owner*. |
| FMT_MSA.1 | Security Management | ICP, AuS | |
| FMT_MSA.3 | Security Management | ICP, ESG, ESC | |
| FMT_SMF.1 | Security Management | ICP | |
| FMT_SMR.1 | Security Management | ICP | |
| | | | |
| FPT_STM.1 | Audit | (OE) | The TOE uses and relies on the OE's capability to provide reliable time stamps. |
| | | | |
| FTP_ITC.1 | Trusted Channel | ECP | Uses FCS_TLS_EXT.1. |

# 8 Extended Components Definition

## 8.1 FCS_CKM_EXT.1 Cryptographic key storage

113. The FCS_CKM_EXT.1 component is a new component of the FCS cryptographic key management family. It requires the TSF to securely store specified secret or private keys that it generates or handles, i.e. to store these keys such that they can be accessed by, or exported to, authorised users only.[41]

114. This component, if present, becomes a new dependency of FCS_CKM.1 Cryptographic key generation.

115. There are no management activities or auditable events foreseen.

116. In FCS_CKM_EXT.1.1, the ST author should specify:

1) The type(s) of key that this component relates to;

2) A well-known key storage method name or a brief description of the method;

3) The assigned standard(s) that document that method. The assigned standard may comprise none, one or more recognised standards publications.

117. **FCS_CKM_EXT.1** Cryptographic key storage
Hierarchical to: No other components
Dependencies: FCS_CKM.1 Cryptographic key generation.
**FCS_CKM_EXT.1.1** The TSF shall store [**assignment**: type(s) of key] cryptographic keys in accordance with a specified cryptographic key storage method [**assignment**: method] that meets the following: [**assignment**: list of standards].

## 8.2 FCS_RBG_EXT.1 Random bit generator

118. The FCS_RBG_EXT.1 component is a new family in the FCS cryptographic support class. It has a single component, FCS_RBG_EXT.1 Random bit generator, which requires the TSF to be capable of generating random bits/numbers in accordance with a recognised standard.

119. This component becomes a new dependency of any FCS components that require random numbers to be generated as part of their implementation.

120. There are no management activities or auditable events foreseen.

121. In FCS_RBG_EXT.1.1, the ST author should specify:

1) One or more recognised standards publications.

[41] In this ST, this component will be specified solely for email decryption keys; and access to and export of these keys will be covered by selected FDP User data protection components.

122. In FCS_RBG_EXT.1.2, the ST author should specify:

1) The source of the entropy, e.g. a software-based noise source or a hardware-based noise source;

2) Either 128 bits or 256 bits of entropy.

123. **FCS_RBG_EXT.1** Random bit generation
Hierarchical to: No other components
Dependencies: No dependencies.
**FCS_RBG_EXT.1.1** The TSF shall implement a deterministic random bit generator in accordance with [**assignment**: list of standards].
**FCS_RBG_EXT.1.2** The deterministic random bit generator shall be seeded by an entropy source that accumulates entropy from [**assignment**: entropy source] with a minimum of [**selection**: 128 bits, 256 bits] of entropy which is at least equal to the greatest security strength required for the keys and hashes that the TSF will generate.

Application note: Security strength should be as defined in the standard(s) assigned in FCS_RBG_EXT.1.1.

## 8.3 FCS_TLS_EXT.1 TLS protocol

124. The FCS_TLS_EXT.1 component is a new family in the FCS cryptographic support class. It has a single component, FCS_TLS_EXT.1 TLS protocol, which requires the TSF to implement the TLS 1.2 protocol, and to validate any X.509 certificates presented to it in the course of establishing a TLS channel with an external entity. (This is the protocol used by the TOE to protect decryption keys being sent to another instance of the TOE.)

125. This component becomes a new dependency of FTP_ITC.1.

126. There are no management activities foreseen.

127. For FCS_TLS_EXT.1, the following actions should be auditable if FAU_GEN Security audit data generation is included in the ST:

1) Establishment (successful or failed) of a TLS channel between the TOE and another instance of the TOE.

128. In FCS_TLS_EXT.1, the ST author should specify:

1) One or more ciphersuite(s) that are acceptable to the TOE. (The TOE should reject any attempt to establish a TLS channel with it using any other ciphersuite.)

129. **FCS_TLS_EXT.1** TLS protocol
Hierarchical to: No other components
Dependencies: FCS_CKM.1 (Cryptographic key generation)
FCS_COP.1 (Cryptographic operation).
**FCS_TLS_EXT.1.1** The TSF shall implement the TLS 1.2 protocol in accordance with RFC 5246 using only the following ciphersuites: [**assignment**: list of ciphersuites].
**FCS_TLS_EXT.1.2** The TSF shall validate any X.509 certificate presented to it in the course of establishing a TLS channel with an external entity in accordance with the following rules: [**assignment**: list of rules].