Security Target for Dencrypt Connex version 6.0

ST Version 0.14

Executive summary

This document is the Common Criteria Security Target for Dencrypt Connex for the iPhone. It is following the specification given in Part 1 annex A of the Common Criteria version 3.1 Revision 5.

Contents

1 Introduction

1.1 Security Target identification and organisation

Title:	Security Target for Dencrypt Connex version 6.0
ST Version:	0.14
Status:	Released
Date:	2021-03-31
Sponsor:	Dencrypt A/S
Developer:	Dencrypt A/S
Keywords:	Mobile application, VoIP, voice and message encryption

This Security Target (ST) has been structured in accordance with [CC] Part 1. The main sections of the ST are the introduction, security problem definition, security objectives, security requirements, TOE summary description and annexes.

The introduction provides general information about the TOE, serves as an aid to understand the nature of the TOE and its security functionality and provide context for the evaluation.

The security problem definition describes the security aspects of the environment in which the TOE is to be used and the manner in which it is to be deployed. The TOE security environment includes:

a) assumptions regarding the TOE's intended usage and environment of use

b) threats relevant to secure TOE operation

c) organisational security policies with which the TOE must comply

The security objectives reflect the stated intent of the ST. They pertain to how the TOE will counter identified threats and how it will cover identified organisational security policies and assumptions. The security objectives are divided into security objectives for the TOE and for the environment. The security objectives rationale demonstrates that the stated security objectives are traceable to all of the aspects identified in the TOE security problem definition and that they are suitable to cover them.

The extended components section identifies any extended security requirements, i.e. requirements that in addition to requirements defined in CC Part 2 and 3 are used within this ST.

The security requirements section provides detailed requirements, in separate subsections, for the TOE and its environment. The security requirements are further divided into the TOE security functional requirements and the TOE security assurance requirements.

The TOE summary specification addresses the security functions that are represented by the TOE to answer the security requirements.

The annex contains a list of abbreviations and a glossary relevant for this ST.

1.2 TOE identification

The TOE is Dencrypt Connex, version 6.0 for Apple iOS. The complete build version is 6.0.0.4.

1.3 TOE type

The TOE is an application for iPhone that offers encrypted mobile voice, video and message communication within well-defined user groups. Once installed and configured, it allows persons to talk or message securely, as well as allowing group calls with more than two persons. Although the application is available for both iPhone and Android, only the iPhone version is considered to be the TOE and evaluated.

1.4 TOE overview

Dencrypt Connex is a component in the Dencrypt Communication Solution. The TOE is an App, running on an iPhone. It is a VoIP client providing end-to-end encrypted voice, video and message communication between iPhones.

The main security features of the TOE and its operational environment are:

- Encrypted end-to-end voice and video communication (Secure Call)
- Encrypted messages (Secure Messaging)
- Encrypted group calls
- Secure Individual phone book
    - Centrally managed (TOE environment)
    - Distributed seamlessly to user devices
    - Supports individual groups settings
    - Supports individual emergency contacts
- Encrypted communication is restricted to administrator defined groups
- Supports secure provisioning to set up a new DCA installation
- Supports its own key-pair generation
- Secure data-at-rest storage of credentials and data
- Encrypted push notifications
- TCP tunnelling for voice or video communication

The Dencrypt Communication Solution consists of Dencrypt Connex (the TOE) and the Dencrypt Server System, which contains: a Dencrypt Communication Server (a SIP and LIMEv2 server), a Dencrypt Database Server (provides database services to servers), a Dencrypt Certificate Manager (signs server and client certificates), a Dencrypt Provisioning Server (provisions clients), Dencrypt Control Center (provides administrator interface) and a Dencrypt Server Bridge (connects to other Dencrypt Server Systems). Only the Dencrypt Connex is part of the TOE. The other parts are not within the scope of the TOE, but are considered as necessary parts of the TOE environment. The Dencrypt Server System is specified in another Security Target and subject to a separate evaluation and certification.

1.5 TOE description

1.5.1 Introduction and intended use

The key feature of Dencrypt Connex and the Dencrypt Communication Solution is to provide mobile devices with end-to-end encrypted voice, video (Secure Call) and message communication (Secure Messaging) within closed user groups that are centrally managed. The following picture displays how the components of the Dencrypt Communication Solution interact.

1.5.2 The TOE architecture and key functions

1.5.2.1 Introduction

The functionalities of the main components of the Dencrypt Communication Solution are described in more details below. It includes both Dencrypt Connex and the Dencrypt Server System (DSS):

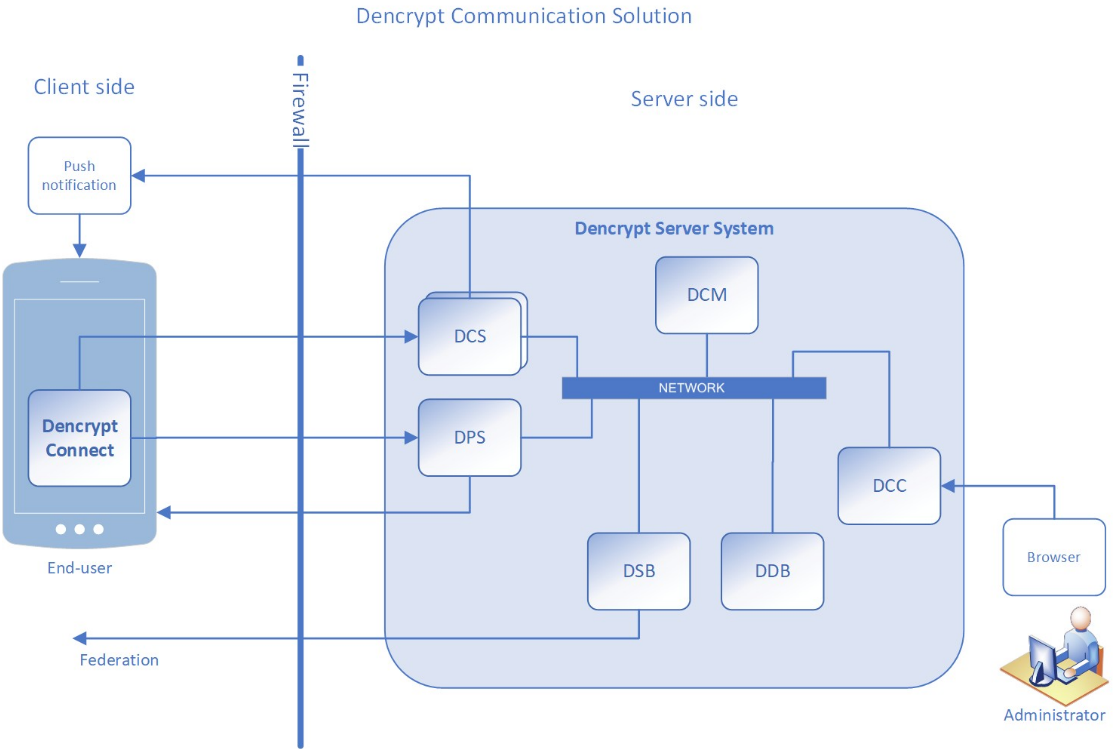

Illustration 1: Dencrypt Communication Solution overview

Dencrypt Connex Application (TOE)

The Dencrypt Connex Application (DCA) is a mobile SIP and messaging client that runs on a mobile device (e.g. an iPhone). The client is able to establish encrypted calls and send encrypted messages between clients on other mobile devices using the SIP and LIMEv2 servers of the Dencrypt Communication Server. The client is installed and updated using the Apple App Store. The client must be configured and initialised before being used. This is done using the provisioning service. The Mobile Client is the TOE.

Dencrypt Provisioning Server (TOE environment)

The Dencrypt Provisioning Server (DPS) is used to initialise clients with user credentials, Dencrypt Communication Server URL and responds to the Certificate Signing Request (CSR) sent from the TOE to set up its certificate. During provisioning, the client is provided with a HTTPS web link for the initialisation. This link can also be encoded into a QR code which the application can scan. The link is provided in a secure way as part of the TOE environment. The HTTPS web link points to the web server of the DPS, which is authenticated by the client.

Dencrypt Communication Server (TOE environment)

The Dencrypt Communication Server (DCS) provides the SIP and LIMEv2 Services that are necessary for the Clients to establish communication between clients.

- SIP Server – necessary for the clients to establish voice and video communication between two or more clients. Also provides routing for messaging.
- LIMEv2 server – provides the key-exchange functionality to initiate secure message communication.
- Tunnel server – provides a server to tunnel voice or video communication over TCP.

Dencrypt Database (TOE environment)

The Dencrypt Database (DDB) provides the database services for the DCS. It keeps the user data and most meta data e.g. call statistics.

Dencrypt Control Center (TOE environment)

The user management is performed using the Dencrypt Control Center (DCC). The user management means creating/deleting users and groups, as well as adding and removing users from these groups. The DCC offers a web interface that is accessible using a web browser from the administrator's local machine.

Dencrypt Certificate Manager (TOE environment)

Dencrypt Certificate Manager (DCM) is the central point for TLS certificates in the system. Once provisioning has taken place, all connections between the DCA and server components use mutually authenticated TLS connections. The required TLS certificates are issued by the DCM by the following procedure: The client or server generates the private/public key pair and creates a certificate signing request (CSR). The CSR is sent to the DCM which signs the CSR if permitted. The DCM provides the certificate back to client/server for employment. All communication between the DCA and the DCM take place via the DCS, except during provisioning where it takes place via the DPS.

Dencrypt Server Bridge (TOE environment)

The Dencrypt Server Bridge (DSB) introduces functionality to make Dencrypt Calls between users on two different DSS. It enables Secure Phonebook synchronization between systems and routing of SIP or messaging data between systems. The connection between two DSS are protected by a TLS channel. The DSB ensure that this connection is authenticated.

1.5.2.2 Provisioning and user registration process

The provisioning process consists of two independent steps:

- Installation of the DCA (the TOE)
- Provisioning of the data to the TOE where the data are the following:
  - DCS user credentials,
  - DSS domain,
  - Signed certificate from the CSR

The DCA is provided and installed using Apple’s App Store. The App Store delivery mechanism operated by Apple is assumed to be trusted and secure. The DCA can also be provided using Apple’s Volume Purchase Program (VPP), where the app is installed via an MDM.

Note that the DCA is delivered in an unprovisioned state. One possibility is to use an MDM to configure apps, however, MDMs usually configure apps with constant values such as user names or server URLs. The DCA provisioning uses one-time tokens, which are by default not constant. Additionally, there might be a separation of duties between MDM administrators and DCA administrators. Thus, the Dencrypt Communication Solution provides its own provisioning server to facilitate the initial configuration of the DCA.

Provisioning is started by the Dencrypt administrator by adding the user to the Dencrypt Communication Solution and directory. After that the administrator will send an invitation message e.g. by email to the user's handset. The invitation message has a link to the web server the user shall tap the link which starts the TOE. The link can also be encoded in a QR code which the phone can scan to load the link into the TOE. The TOE parses the link, fetches the provisioning data from the DPS and installs the data. The provisioning data are deleted on the DPS, i.e. the HTTPS link can be used only once. Additionally, the link is only valid for a limited time after the link has been provided. The URL, i.e. the token in the URL, is verified by the server when the client connects. The URL check is implemented in the TLS module and a token check failure results in a TLS connection termination. Thus, Dencrypt describes the DPS TLS connection as token-based client authenticated. This feature allows to connect securely to the DPS from any internet connection and securely provision a new device.

The settings and phone book of the DCA (TOE) are updated as described in chapter "Managing settings and phone book". After that the TOE is setup and operational.

In addition to the phonebook, other DCA settings are also controlled by the DSS. For instance, the ability to integrate with native iOS call history. The DSS also assists the TOE in protecting data-at-rest by storing an encrypted storage key of the TOE. This key is submitted at the end of provisioning. The TOE must connect to the DSS, retrieve and decrypt this key to be able to decrypt any data stored outside of RAM. For subsequent use of the DCA, this key is downloaded from the DSS before files stored by the DCA can be accessed.

1.5.2.3 Managing settings and phone book

The DCA only allows calls and messaging to persons listed in the DCA phone book as well as to predefined emergency contacts. The phone book is individual for each user and contains only the persons which a user is allowed to call. Thus, each user may have a different phone book. Note you may be able to receive calls from users not in your phone book. The user administrator (TOE environment) can change the groups of users to whom a specific user can call to at any time. The administrator also has the ability to send out user notifications to one or more DCAs regardless of the phonebooks. Users of the DCA have the ability to call users of a different DSS if that system is connected via the DSB. The administrator of a DSS can then push phonebooks to the remote system.

The DCS (also TOE environment) takes care of distributing the phone book to the individual TOE users. When a user starts the TOE, the TOE establishes a TLS connection to the DCS and makes a SIP registration. When registration is successful, the client will send a web request to check if the phonebook or settings have changed. The client will then regularly send these web requests. If the phone book has been changed, DCS notifies the client about the current phone book version. The client downloads the phone book if its currently used phone book version does not match the advertised phone book version. Note, if the client has no phone book, it is considered as phone book version 0. The same method applies for settings distribution. The following figure displays the described process.

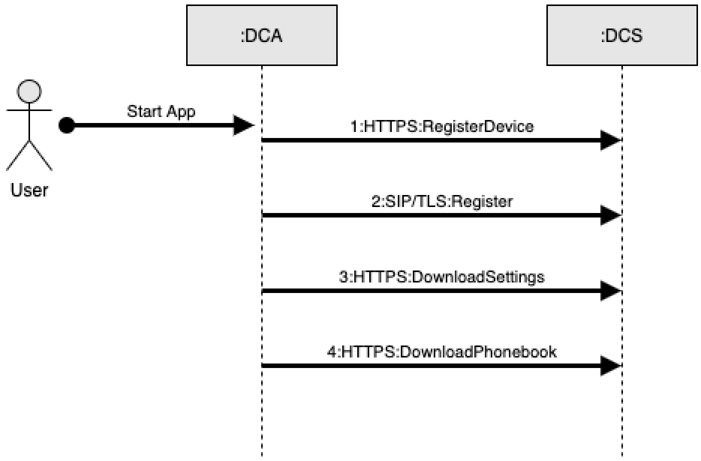

Illustration 2: Registration process

1.5.2.4 Making a secure call

The uniqueness of this TOE is that the end-to-end encrypted voice, video and messaging use Dynamic Encryption, which ensures that each call session is encrypted with an additional layer using a randomly chosen algorithm parameters (S-boxes) and randomly chosen keys.

The following figure illustrates the steps for a secure call between two DCAs.

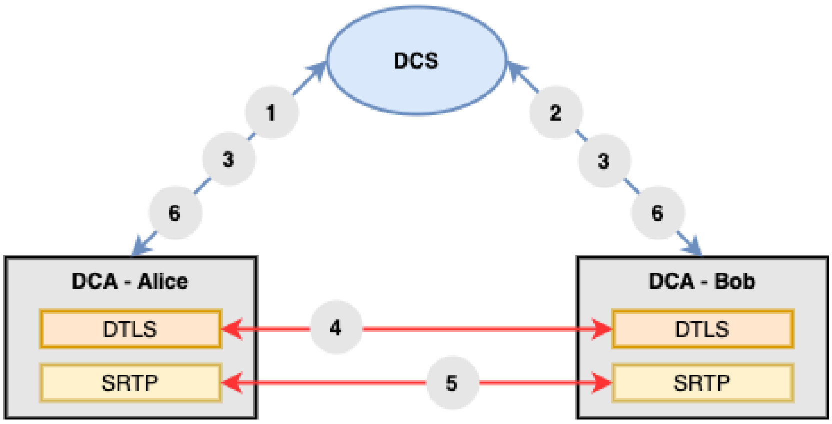

Illustration 3: Secure voice call description

1. Alice's DCA contacts the DCS she is registered to.
2. The SIP server resolves Bob's address and contacts Bob's DCA. This resolution is limited because Alice can only contact the DCS for users listed in her phone book.
   Note: For calls to users of a different DSS, these are forwarded via the DCS. When a call is received to a user outside of the current system, the call establishment (SIP) is forwarded to the DCS of a remote system from the DCS of the TOE over a TLS connection.
3. SIP takes care of signalling, i.e. triggers Bob's DCA to start ringing. As soon as Bob accepts the call, both DCA are signalled to start a media session for the real-time audio data stream.
4. Before the audio connection is encrypted, DTLS-SRTP takes over the data of media session. DTLS-SRTP exchanges certificates and fingerprints to authenticate the callers. A shared secret is then negotiated between Alice and Bob's DCA. Additionally, DTLS-SRTP has been modified to securely negotiate the ciphersuite to be used for Dynamic Encryption. DTLS-SRTP also authenticates the caller certificates.
   Note: The TOE has the ability to provide a tunnelling service to tunnel all encrypted voice and video communication data over TCP and TLS. Note that this tunnel is not used for security purposes as the tunnelled call audio data is still end-to-end encrypted.
5. Once DTLS-SRTP has established the shared secret, it calculates different keys for the bi-directional audio data stream between Alice and Bob. The key and Dynamic Encryption parameters derived using Key Boosting. These are required for the dynamic encryption of the audio data stream. The dynamically encrypted real-time data is transported over the IP network by the secure variant of the real-time protocol, so called SRTP.
6. When Bob ends the call the DCS signals the call termination to Alice. All key material are erased.

The following list characterises the secure call in the TOE:

- Dynamic encryption of communication data is implemented as multiple layers of encryption optimized for voice or video data over the SRTP protocol.
- The voice or video stream is bidirectional, i.e. each direction uses different en/decryption keys.

- The DCA completes a DTLS handshake over ECDHE to initiate the secure channel.
- DTLS provides the shared secret to SRTP-KDF initiating the end-to-end encrypted communication.
- To mitigate a man-in-the-middle attack of the media channel, the callers send their certificates over the media stream and the certificates' fingerprints over the SIP signalling channel. Each DTLS-SRTP endpoint verifies that the fingerprint matches the certificate.
- To authenticate the callers after call setup, each caller will provide the SIP ID and the certificate of the remote party to the server. The server verifies that the SIP ID is associated with that certificate.
- Dynamic Encryption for voice and video uses the following keys, which are derived from the DTLS shared secret using Key Boosting:
  - 256-bit key for the standard AES-256 encryption. This key is provided by DTLS-SRTP.
  - 128-bit Dynamic Encryption algorithm selection key that defines the S-box for an additional AES-round.
- Dynamic Encryption keys and algorithms are established at call setup and destroyed as soon as the call is terminated.
- Random number generation uses the RNG on Apple iOS (TOE environment).

1.5.2.5 Secure Messaging

DCA (TOE) provides secure end-to-end encrypted messaging between two or more end users. Secure messaging enables asynchronous text based communication between the parties, where attachments can also be sent. Messaging is implemented through the LIMEv2 protocol, which in turn is based on the Signal protocol. The DCS (TOE environment) is acting as a trusted server providing identification of peer devices and routing of messages. Note that Secure Messaging is asynchronous, which differs from secure calls which are direct. The server is used to stores messages and public keys until the receiver of a message retrieves them.

The following picture provides an overview of messaging between DCAs and the cryptographic protocols involved.

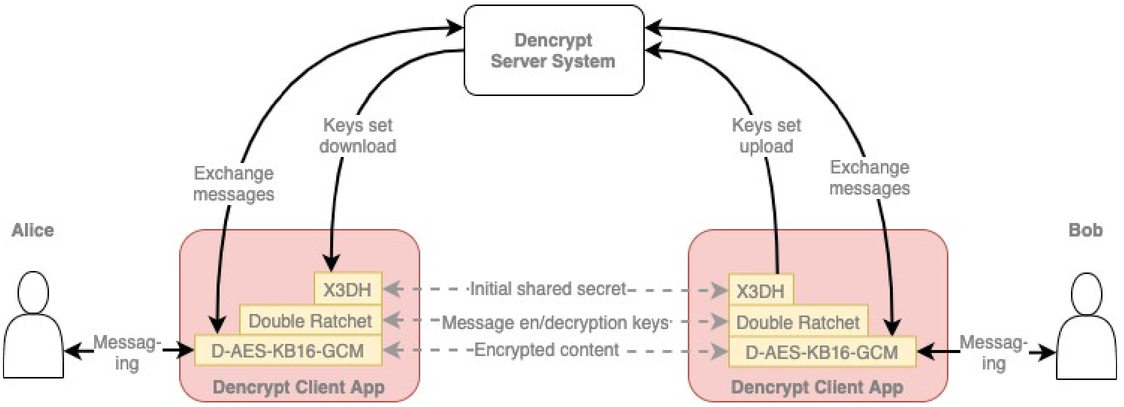

Illustration 4: Secure messaging description

1. Key pairs are generated by the DCAs and the public keys are uploaded to the LIMEv2 server for the X3DH protocol.
2. The sender of a message downloads these keys, generates an ephemeral key pair, and performs the following operations: Diffie-Hellman computations generates shared keys that are fed into a Double Ratchet KDF, which in turn generates a chain key. The chain key is fed through the KDF to compute a message key and a new chain key.

3. The 256-bit message key uses Dencrypt Key Boosting to create a 384-bit key that is used for Dynamic Encryption.
4. The encrypted message is sent over SIPS together with the sender's public identity keys and DH public keys.
5. The receiver will use the keys to perform the same key derivation process and decrypt the message.
   Note: After having established a shared secret once, only Double Ratchet is used to derivate keys and not X3DH.

1.5.2.6 The TLS connection

All connections made between the TOE and any other component, including the one with the SIP server, are established using a trusted channel that is implemented using TLS version 1.2. The exceptions are the VoIP connection and messages sent between the TOE and another instance of the DCA. The secure messages are encrypted separately via the LIMEv2 protocol, while voice and video is encrypted via the DTLS-SRTP and SRTP protocols. The connections are encrypted using Dynamic Encryption as described in the previous sections.

Note that all TLS connections are initiated by the TOE and never by the server backend, such as the SIP server. The TOE is keeping the SIPS TLS connection alive as long as the TOE is running. It is necessary for being able make and receive calls and messages.

Push notifications sent from the server to the DCA are not sent through this TLS channel, but instead transmitted via the mobile operating system vendor's (in this case Apple) systems. Its sensitive contents are encrypted by AES and decrypted within the application once it has started.

The TLS connection is mutually authenticated to ensure both that only authorized instances of the TOE can establish connections, i.e. to retrieve the phone book information, and that the TOE is not connecting to a server system that may deceive the TOE user with false phone books or provisioning data. The TLS connection also ensures the confidentiality and integrity of any data transmitted. The TLS connection is always initiated by the TOE and never by the DCS or DPS.

Please note that the TLS connection to the provisioning web server is not mutually authenticated because the TOE is not yet configured and has no signed client certificate.

Finally, a TLS connection can also be used to tunnel the encrypted voice or video communication over TCP. If SRTP communication is not possible over the networks connecting two DCA users, the TOE can provide a TCP tunnel for said communication. The tunnel does not provide the protection of the traffic, since it is still end-to-end encrypted via SRTP.

Details of the protocols and cipher suites used are provided in the TOE Summary Specification in chapter 7.

1.5.2.7 Platform security

The DCA utilizes the available security mechanisms of the underlying platform, i.e. the iPhone device and iOS operating system. Authentication via Apple Touch ID or Face ID is required during start-up. In addition to these mechanisms, the TOE also implements an additional layer of data-at-rest protection to protect stored application data or files. There are two keys involved in this process:

- one Data Encryption Key called Storage Key (which encrypts the stored data)
- one Key Encryption Key called Device Key (which encrypts the previously mentioned Storage Key).

All data stored on the device is encrypted by the Storage Key, which is only persistently stored on the DSS. This key is encrypted by the Device Key before being uploaded to the DSS. The DCA only stores the Device Key locally. This means that the DCA needs to connect and authenticate to the DSS to access the any stored files. If a user is removed on the DSS then the key, and therefore also the files, are not accessible. DCA data and messages are stored in an encrypted local database, while file attachments from messaging are stored encrypted in the file system.

The application also employs further safeguards against mismanagement of the application data by the platform. Restoring from an iCloud backup or quick migration is not possible in case the phone is lost. The application does not support any backup or export of messaging history. The DSS can also enable settings to disable native iOS interactions (e.g. camera or external viewers for attachments). Additionally, the iOS Keychain protection by the Secure Enclave protects the private TLS client key, the Device Key to encrypt the Storage Key, and the mobile push encryption key.

1.5.3 Security functions

This section provides a summary of the security functions implemented by the TOE. These security functions have previously been described in the previous section.

- Encrypted end-to-end voice and video communication (Secure Call)
- Encrypted messages (Secure Messaging)
- Encrypted group calls
- Secure Individual phone book
    - Centrally managed (TOE environment)
    - Distributed seamlessly to user devices
    - Supports individual groups settings
    - Supports individual emergency contacts
- Encrypted communication is restricted to administrator defined groups
- Supports secure provisioning to set up a new DCA installation
- Supports its own key-pair generation
- Secure data-at-rest storage of credentials and data
- Encrypted push notifications
- TCP tunnelling for voice or video communication

1.5.4 Physical scope of the TOE

The TOE is limited to the DCA and user documentation. The following documentation is provided to the users:

- *Operational User Guide DCA v. 6.0 (iOS)*
- *Preparative Guide DCA v. 6.0 (iOS)*

The TOE is delivered to the user via the operating system's built-in software distribution system, i.e. the Apple App Store. It can also be delivered via Apple's VPP using an MDM. Note that both the operating system and the App Store are outside the scope of the TOE. The developer will upload the DCA to the Apple App Store, where a customer can find and download it using an Apple ID account. The user installs and configures the app by following the instructions given in the user documentation. Since Apple will receive and approve the DCA before it is made available on the App Store, Apple is seen as a trusted middle-man for the delivery process.

1.5.4.1 IT environment

The TOE environment consists of the IT components that make up the mobile devices but are outside of the TOE as well as any IT components that are outside of mobile device.

The IT components that are outside of the TOE but part of the mobile devices are:

- The mobile device hardware (iPhone) and iOS software on which the TOE is installed. This includes the App Store application included in iOS, which is used for the delivery mechanism.

The IT environment must contain the following:

- The mobile device (iPhone) where the TOE is installed.
- Any additional mobile devices where the TOE is also installed (to have another party to securely communicate with)
- The DSS with DPS, DCC, DCM, DDB, DSB and DCS, as well as any standard MDM system in the case that the DCA is distributed using Apple's VPP.

The DSS must be used and operated by administrators that are trustworthy and have been sufficiently trained to use and carry out the security management tasks in a proficient manner. The TOE users must be trustworthy and trained and are expected to follow instructions.

2 Conformance claims

2.1 CC conformance claim

This ST is CC Part 2 extended and CC Part 3 conformant. This ST claims conformance to CC version 3.1 Revision 5, April 2017.

This ST claims conformance to the EAL4 package of security assurance requirements, augmented with ALC_FLR.2. This ST does not claim conformance to any Protection Profile (PP).

2.2 Conformance rationale

In general, assurance requirements must be commensurate with the exposure of systems to untrustworthy and unauthorized entities. For example, mobile devices will be more exposed to attackers than systems in a well-guarded environment, but exposure through communication channels may jeopardize even systems guarded in secure vaults.

Since the architecture addressed by the TOE specified in this ST includes systems where both the attack potential and the value of the assets are likely to be high, a sufficient level of assurance must be selected to provide system users with appropriate assurance that the system will be able to withstand such threats.

The TOE is expected to provide assurance to ensure separation of compartments, which requires a level of assurance that includes the evaluation of side channels between different compartments.

The EAL4 level was also deemed appropriate because this will provide a necessary assurance for encrypted communication services such as secure voice, secure video and secure message.

3 Security problem definition

A mobile device may be used in different ways. A device, where the TOE is installed, must be under significant enterprise control over the configuration and software inventory. The enterprise elects to provide users with mobile devices and control the configuration as well as the set of applications that can be installed in order to maintain a high degree of control of their enterprise data and security of their networks.

It is assumed that the TOE is under physical control of the user and that the users are trained and trusted to handle the TOE and to access to the enterprise data and services they are given access to. Although the users are assumed to be trustworthy and trained, we cannot exclude that mistakes are being made.

3.1 Threats

This section of the security problem definition describes the threats that are countered by the TOE, its operational environment, or a combination of the two.

Threat agents are typically characterized by a number of factors such as expertise, available resources, and motivation, with the motivation being linked directly to the value of the assets at stake.

Threat agents are entities such as unauthorized individuals or authorized users that are trying to act outside of their authorization or any other entities acting on behalf of unauthorized users, such as users. Those may attempt to get access to TSF services either by masquerading as an authorized entity or by attempting to use TSF services without proper authorization.

The term *threat agent* is used to indicate that a threat can be performed by an unauthorized external entity, an authorized external entity or an untrusted app. Threat agents are assumed to have moderate level of expertise, resources and motivation.

The following threats are addressed by the TOE and the TOE environment.

Threat	Description
T.DATA	An unauthorized user or attacker will gain access to user credentials, TOE settings or phone book entries to which they are not authorized. This involves data sent between the TOE and the DSS, including push notifications.
T.MASQUERADE	A user within a closed user group is masquerading, pretending to be another user to mislead the receiver that a secure voice or video call, or a secure message is originating from another user belonging to the phone book of that user group.
T.TRAFFIC	An attacker (including network operators) may gain access (disclosure or modification) to secure voice, video or messaging conversations between users within a closed user group.

3.2 Organisational security policies

The following organisational security policies are enforced by the TOE and the TOE environment.

OSP	Description
OSP.CLOSED	The TOE shall ensure that secure calls and secure messages are restricted to parties defined by the phone book on the TOE of the calling party.

OSP	Description
OSP.FORWARD	The TOE must be able to prevent an unauthorized user that obtains a handset to decrypt previously transmitted traffic (voice, video or message) that has been encrypted using the obtained handset. Any encryption keys used for the transmitted traffic must be erased.
OSP.PRIVATEKEY	The TOE must be able to generate its own private-public key pairs.
OSP.MANAGE	The TOE shall allow secure provisioning and remote update of certificates and phone book.
OSP.PHONEBOOK	The TOE must ensure that the phone book cannot be changed locally.
OSP.UPTODATE	The TOE must ensure that the phone book held by the TOE is up-to-date.
OSP.STORAGE	The TOE must use cryptography to protect the confidentiality of data stored on the device. The data involves both DCA data, messages and message attachments. The data must only be accessible if the user can authenticate to the DSS.
OSP.TUNNEL	The TOE must support tunnelling of voice or video communication over TCP by tunnelling the connection over TLS 1.2.

3.3 Assumptions

This section specifies the assumptions on the TOE environment that are necessary for the TOE to meet its security objectives.

Assumption	Description
A.ADMIN	It is assumed that the TOE administrators (i.e. the administrators using the DSS) are trustworthy and trained to perform the actions required by them for the management and maintenance of the DSS.
A.APPS	It is assumed that only approved, benign applications are running on the handset where the TOE is running.
A.BACKEND	It is assumed that the underlying hardware, firmware (BIOS and device drivers) and software of the server system used by the TOE are working correctly and have no undocumented security critical side effect on the TOE. Furthermore, the server system is operated in a physically secure and well managed environment.
A.HANDSET	It is assumed that the functions in the TOE environment related to memory management, program execution, access control and privilege management provided by the underlying iOS of the handset and the SIM card, work correctly and have no undocumented security critical side effects on the security functions of the TOE.
A.KEYS	It is assumed that random bits provided by the underlying platform are of good quality and have sufficient entropy.
A.SINGLEUSER	It is assumed that the TOE is under the physical control of a single authorized user.
A.USER	It is assumed that the users are trustworthy and trained to perform their actions in accordance with their instructions and security policies.

Assumption	**Description**
A.PROVISIONING	It is assumed that the operational environment ensures that the web link is not predictable, only active for a limited time and that access to the link is limited to one attempt only. It is also assumed that the operational environment provides the link to clients in a secure way so that the link is not disclosed to any potential attacker. Note: The link might be disclosed for the user's organisation, e.g. the link might be in cleartext on the organisation's local mail server.
A.APPSTORE	It is assumed that the operational environment provides a secure delivery mechanism for the TOE operated by a trusted third party. For the TOE which runs on the iPhone iOS operating system, this is the built in App Store operated by Apple.

4 Security objectives

The security objectives provide a concise statement of the intended response to the security problem. It will describe which security needs will be addressed by the TOE and which will be addressed by the TOE environment, in the form of a statement of security objectives.

4.1 Security objectives for the TOE

The following are the security objectives to be met by the TOE.

Security Objective	Description
O.CALLERID	The TOE must ensure that the end point of a secure call connection or a secure message is unique and that the caller display name associated with the caller identity is correctly shown to the TOE user making or receiving the secure call or message.
O.GROUP	The TOE must ensure that secure calls and messages are restricted to users within the TOE phone book of the calling party.
O.TRAFFIC	The TOE must ensure that secure calls are protected against disclosure and modification.
O.MESSAGES	The TOE must ensure that secure messages are protected against disclosure and modification.
O.CHANNEL	The TOE must ensure that there is a trusted path between the TOE and the DSS ensuring authenticity, confidentiality and integrity of any TSF or user data transmitted between the TOE and the server system, such as phone book updates and SIP connections made when establishing secure calls.
O.PHONEBOOK	The TOE must ensure that the phone book cannot be changed locally.
O.FORWARD	The TOE must ensure that an unauthorized user that obtains a handset cannot decrypt previously transmitted traffic (voice, video or messages) that has been encrypted using the obtained handset. Any encryption keys used for the transmitted traffic must be erased.
O.PRIVATEKEY	The TOE must be able to generate its own private-public key pairs.
O.MANAGE	The TOE must support provisioning and ensure that settings and phone book can be updated whenever the TOE is running and registered on the DCS.
O.STORAGE	The TOE must use cryptography to protect the confidentiality of data stored on the device. The data involves both DCA data, messages and file attachments. The data must only be accessible if the user can authenticate to the DSS.
O.TUNNEL	The TOE must support tunnelling of voice or video communication to the server over TCP by tunnelling the connection over TLS 1.2.
O.PUSH	The TOE must be able to decrypt Push Notifications that are sent via the mobile operating system vendor’s systems.

4.2 Security objectives for the TOE environment

The following are the security objectives to be met by the TOE environment.

Security Objective	Description
OE.ADMIN	The operational environment shall ensure that the TOE administrators (i.e. the administrators using the server system) are trustworthy and trained to perform the actions required by them for the management and maintenance of the DSS.
OE.APPS	The operational environment shall ensure that only approved, benign applications are running on the handset where the TOE is running.
OE.BACKEND	The operational environment shall ensure that the underlying hardware, firmware (BIOS and device drivers) and software of the DSS system used by the TOE are working correctly and have no undocumented security critical side effect on the TOE. The operational environment shall also ensure that the server system is operated in a physically secure and well managed environment.
OE.HANDSET	The operational environment shall ensure that the functions in the TOE environment related to memory management, program execution, access control and privilege management provided by the underlying iOS of the handset and the SIM card, work correctly and have no undocumented security critical side effects on the security functions of the TOE.
OE.KEYS	The operational environment shall ensure that random bits provided by the underlying platform are of good quality and have sufficient entropy.
OE.SINGLEUSER	The operational environment shall ensure that the TOE is under the physical control of a single authorized user.
OE.USER	The operational environment shall ensure that the users are trustworthy and trained to perform their actions in accordance with their instructions and security policies.
OE.PROVISIONING	The operational environment shall ensure that the web link is not predictable, only active for a limited time and that access to the link is limited to one attempt only. It shall also provide the link to clients in a secure way so that the link is not disclosed to any potential attacker. Note: The link might be disclosed for the user's organisation, e.g. the link might be in cleartext on the organisation's local mail server.
OE.APPSTORE	The operational environment shall provide a secure delivery mechanism for the TOE operated by a trusted third party. For the TOE which runs on the iPhone iOS operating system, this is the built in App Store operated by Apple.

4.3 Security objectives rationale

4.3.1 Security objectives completeness

The following tables provide a mapping of security objectives both for the TOE and the TOE environment to the environment defined by the threats, policies and assumptions, illustrating that each security objective for the TOE covers at least one threat or policy, and that each security objective for the TOE environment covers at least one policy, threat or assumption.

	T.DATA	T.MASQUERADE	T.TRAFFIC	OSP.CLOSED	OSP.FORWARD	OSP.PRIVATEKEY	OSP.MANAGE	OSP.PHONEBOOK	OSP.UPTODATE	OSP.STORAGE	OSP.TUNNEL	A.ADMIN	A.APPS	A.BACKEND	A.HANDSET	A.KEYS	A.SINGLEUSERS	A.USER	A.PROVISIONING	A.APPSTORE
O.CALLERID		X																		
O.GROUP				X																
O.TRAFFIC			X																	
O.MESSAGES			X																	
O.CHANNEL	X	X																		
O.PHONEBOOK				X				X												
O.FORWARD					X															
O.PRIVATEKEY						X														
O.MANAGE							X		X											
O.STORAGE										X										
O.TUNNEL											X									
O.PUSH	X																			
OE.ADMIN												X								
OE.APPS													X							
OE.BACKEND														X						
OE.HANDSET															X					
OE.KEYS						X										X				
OE.SINGLEUSER																	X			
OE.USER																		X		
OE.PROVISIONING							X												X	
OE.APPSTORE																				X

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	This threat is addressed by O.CHANNEL that ensures that there is a trusted path between the TOE and the server system ensuring authenticity, confidentiality and integrity of any TSF or user data transmitted between the TOE and the server system, such as phone book updates and SIP connections made when establishing secure calls. O.PUSH ensures that Push Notifications containing user data sent form the DSS to the TOE are encrypted. These are transmitted via the mobile device operating system vendor.

Threat	Rationale for the security objectives
T.MASQUERADE	This threat is addressed by O.CHANNEL that enforces client authentication and by O.CALLERID that ensures that the end point of a secure call or secure message is unique and that the caller display name associated with the caller identity is correctly shown to the TOE user making or receiving the security call or message. T
T.TRAFFIC	This threat is addressed by O.TRAFFIC and O.MESSAGES. O.TRAFFIC ensures that secure calls are protected against disclosure and modification, while O.MESSAGES ensures that secure messages are protected against disclosure and modification. Note that these two protection mechanisms both involve communication between two or more DCAs, but O.MESSAGES involve asynchronous secure messaging communication.

The following rationale provides justification that the security objectives of the TOE and the TOE environment are 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 security objectives
OSP.CLOSED	This OSP is addressed by O.GROUP that ensures that secure calls and messages are restricted to users within the TOE phone book of the calling party and by O.PHONEBOOK that ensures that the phone book cannot be changed locally.
OSP.FORWARD	This OSP is addressed by O.FORWARD that ensures that an unauthorized user that obtains a handset cannot decrypt previously transmitted traffic (voice, video or message) that has been encrypted using the obtained handset.
OSP.PRIVATEKEY	This OSP is addressed by O.PRIVATEKEY that ensures that the key pair for the TOE is generated by the TOE. This is supported by OE.KEYS that provides the random number for the key generation.
OSP.MANAGE	This OSP is addressed by O.MANAGE that ensures that settings and phone book can be updated whenever the TOE is running and registered on the DCS. The TOE provides management capabilities and ensuring that they are restricted to authorized subjects. The secure provisioning is supported by OE.PROVISIONING.
OSP.PHONEBOOK	This OSP is addressed by O.PHONEBOOK that ensures that phone book cannot be changed locally.
OSP.UPTODATE	This OSP is addressed by O.MANAGE that ensures the phone book is updated whenever the TOE is running and registered on the DCS.
OSP.STORAGE	This OSP is addressed by O.STORAGE that ensures that any data stored by the TOE must be encrypted and that the decryption key must only be accessible after successful authentication to the server.
OSP.TUNNEL	This OSP is addressed by O.TUNNEL that ensures the TOE can establish a TLS 1.2 tunnel to the server.

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 the security objectives
A.ADMIN	Addressed by OE.ADMIN, which is identical to the assumption

Assumption	**Rationale for the security objectives**
A.APPS	Addressed by OE.APPS, which is identical to the assumption
A.BACKEND	Addressed by OE.BACKEND, which is identical to the assumption
A.HANDSET	Addressed by OE.HANDSET, which is identical to the assumption
A.KEYS	Addressed by OE.KEYS, which is identical to the assumption
A.SINGLEUSER	Addressed by OE.SINGLEUSER, which is identical to the assumption
A.USER	Addressed by OE.USER, which is identical to the assumption
A.PROVISIONING	Addressed by OE.PROVISIONING, which is identical to the assumption
A.APPSTORE	Addressed by OE.APPSTORE, which is identical to the assumption

5 Extended components definition

The extended requirements are used to specify TLS for clients and servers. A TOE that implements TLS must in addition to FTP_ITC.1 or FTP_TRP.1 also specify the TLS protocol that is implemented. This is done in FCS_TLSC_EXT.1 and FCS_TLSC_EXT.2 (for cryptography). FCS_TLSC_EXT.2 also describes the TLS protocol used for the tunnelling service. The extended components defined in this Security Target have been copied from [cPPND].

5.1.1.1 FCS_TLSC_EXT – TLS Client Protocol

Family Behaviour

The component in this family addresses the ability for a client to use TLS to protect data between the client and a server using the TLS protocol.

Component levelling

FCS_TLSC_EXT TLS Client Protocol
1
2

FCS_TLSC_EXT.1 TLS Client requires that the client side of TLS be implemented as specified.

FCS_TLSC_EXT.2 TLS Client requires that the client side of the TLS implementation include mutual authentication.

Management: FCS_TLSC_EXT.1, FCS_TLSC_EXT.2

The following actions could be considered for the management functions in FMT:

a) There are no management activities foreseen.

Audit: FCS_TLSC_EXT.1, FCS_TLSC_EXT.2

The following actions should be considered for audit if FAU_GEN Security audit data generation is included in the PP/ST:

a) Failure of TLS session establishment

b) TLS session establishment

c) TLS session termination

5.1.1.1.1 FCS_TLSC_EXT.1 – TLS Client Protocol

Hierarchical to: No other components

Dependencies: FCS_CKM. 1 Cryptographic Key Generation

FCS_CKM.2 Cryptographic Key Establishment

FCS_COP.1/DataEncryption Cryptographic operation (AES Data encryption/decryption)

FCS_COP.1/SigGen Cryptographic operation (Signature Generation and Verification)

FCS_COP.1/Hash Cryptographic operation (Hash Algorithm)

FCS_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)

FCS_TLSC_EXT.1.1 The TSF shall implement [selection: *TLS 1.2 (RFC 5246)*, *TLS 1.1 (RFC 4346)*] and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites:

- [assignment: *list of optional ciphersuites and reference to RFC in which each is defined*].

Application Note

The ciphersuites to be tested in the evaluated configuration are limited by this requirement.

FCS_TLSC_EXT.1.2 The TSF shall verify that the presented identifier matches the reference identifier per RFC 6125 section 6.

Application Note

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the evaluation activity.

FCS_TLSC_EXT.1.3 When establishing a trusted channel, by default the TSF shall not establish a trusted channel if the server certificate is invalid. The TSF shall also [selection:

- *Not implement any administrator override mechanism*
- *require administrator authorization to establish the connection if the TSF fails to [selection: match the reference identifier, validate certificate path, validate expiration date, determine the revocation status] of the presented server certificate*

].

FCS_TLSC_EXT.1.4 The TSF shall [selection: *not present the Supported Elliptic Curves Extension, present the Supported Elliptic Curves Extension with the following NIST curves: [selection: secp256r1, secp384r1, secp521r1] and no other curves*] in the Client Hello.

Application Note

If ciphersuites with elliptic curves were selected in FCS_TLSC_EXT.1.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in FCS_TLS_EXT.1.1, then "not present the Supported Elliptic Curves Extension" should be selected.

This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1/SigGen and FCS_CKM.1 and FCS_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.

5.1.1.1.2 FCS_TLSC_EXT.2 – TLS Client Protocol with Authentication

Hierarchical to:	FCS_TLSC_EXT.1 TLS Client Protocol
Dependencies:	FCS_CKM.1Cryptographic Key Generation
	FCS_CKM.2 Cryptographic Key Establishment
	FCS_COP.1/DataEncryption Cryptographic operation (AES Data encryption/decryption)
	FCS_COP.1/SigGen Cryptographic operation (Signature Generation and Verification)
	FCS_COP.1/Hash Cryptographic operation (Hash Algorithm)

FCS_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)

FCS_RBG_EXT.1 Random Bit Generation

FCS_TLSC_EXT.2.1 The TSF shall implement [selection: *TLS 1.2 (RFC 5246)*, *TLS 1.1 (RFC 4346)*] and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites:

- [assignment: *list of optional ciphersuites and reference to RFC in which each is defined*].

Application Note

The ciphersuites to be tested in the evaluated configuration are limited by this requirement.

FCS_TLSC_EXT.2.2 The TSF shall verify that the presented identifier matches the reference identifier per RFC 6125 section 6.

Application Note

The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the evaluation activity.

FCS_TLSC_EXT.2.3 When establishing a trusted channel, by default the TSF shall not establish a trusted channel if the server certificate is invalid. The TSF shall also [selection:

- *Not implement any administrator override mechanism*
- *require administrator authorization to establish the connection if the TSF fails to [selection: match the reference identifier, validate certificate path, validate expiration date, determine the revocation status] of the presented server certificate*

].

FCS_TLSC_EXT.2.4 The TSF shall [selection: *not present the Supported Elliptic Curves Extension*, *present the Supported Elliptic Curves Extension with the following NIST curves: [selection: secp256r1*, *secp384r1*, *secp521r1] and no other curves*] in the Client Hello.

Application Note

If ciphersuites with elliptic curves were selected in FCS_TLSC_EXT.1.1, a selection of one or more curves is required. If no ciphersuites with elliptic curves were selected in

FCS_TLS_EXT.1.1, then "not present the Supported Elliptic Curves Extension" should be selected.

This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1/SigGen and FCS_CKM.1 and FCS_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.

FCS_TLSC_EXT.2.5 The TSF shall support mutual authentication using X.509v3 certificates.

6 Security requirements

6.1 Security functional policies

6.1.1 GROUP SFP

The TOE will implement an information flow control policy named **GROUP SFP**.

Policies are used to enforce security guidelines and restrict the users from undesired behaviour. The TOE is implementing the **GROUP** information flow control security functional policy, or simply **GROUP SFP**. This will ensure that the TOE does not allow any secure call or secure messaging between a user and another user, unless the user is explicitly allowed to. This policy is set by the central managed phone book for each user. The phone book on the TOE is kept in sync with server system's phone book whenever the TOE is running and registered on the DCS. Secure Call and Secure Messaging are restricted to parties in the phone book.

6.1.2 ENCRYPTION KEY SFP

The TOE must implement a functional policy for encrypted Storage Keys imported from the DSS. The TOE must be able to receive these keys and use them to decrypt the stored date (i.e. message chat history or file attachments) while also ensuring that the keys themselves are not stored persistently on the device. These keys are received through a mutually authenticated TLS channel, where the DSS is authenticated. This does not involve any information flow or access control policy, and the keys are received securely from a trusted source.

6.2 Security functional requirements

The following convention is used for operations applied to the Security Functional Requirements: Assignment and selection are indicated by **bold**. Refinements are indicated by **bold underscore** for additions and by ~~**bold strike through**~~ for deletions. Iterations are indicated by appending a letter to the requirement, e.g. FCS_COP.1a.

Voice and video encryption

These SFRs describe the cryptographic functionality for secure voice and video.

6.2.1 FCS_CKM.2a – Cryptographic key distribution (DTLS-SRTP)

FCS_CKM.2.1 The TSF shall **perform** ~~**distribute**~~ cryptographic **key establishment** in accordance with a specified cryptographic key **establishment** ~~**distribution**~~ method **DTLS-SRTP using Elliptic Curve Diffie-Hellman** that meets the following: **[RFC5763] and [RFC5764]**.

Application note: This SFR covers the key establishment and distribution for secure voice and video. The DTLS-SRTP key agreement protocol performs a Diffie-Hellman key exchange during call setup in the media path. It generates a shared secret, which is then used to generate keys and salt for a Secure RTP (SRTP) [RFC3711] session via Dynamic Encryption FCS_COP.1d. Note that the SRTP keys will pass through a Key Boosting (hash) algorithm in FCS_COP.1c to derive the keys and salt. This SFR has been refined to specify that it involves key establishment rather than solely distribution.

Application note: The key establishment does not use DTLS to authenticate the remote party. RFC5764 separately describes certificate fingerprint validation over SIP during key establishment and call setup. This validation uses the SHA-512 hash function described in FCS_COP.1c.

Messaging encryption

These SFRs describe the cryptographic functionality for secure messaging.

6.2.2 FCS_CKM.1c – Cryptographic key generation (LIME v2.0 EdDSA)

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **as defined in LIMEv2 protocol** and specified cryptographic key sizes **456-bit EdDSA448** that meet the following: **[LIME], [X3DH], [RFC7748], [RFC8032] and [DR]**.

Application note: These are the keys generated and used by the DCA as part of secure messaging. Note that this is an asymmetric key and involved a private/public key-pair.

6.2.3 FCS_CKM.2c – Cryptographic key distribution (LIME v2.0 establishment)

FCS_CKM.2.1 The TSF shall **perform** ~~**distribute**~~ cryptographic **key establishment** in accordance with a specified cryptographic key **establishment** ~~**distribution**~~ method **as defined in the LIMEv2, X3DH and Double Ratchet specifications using Elliptic Curve Diffie-Hellman** that meets the following:**[LIME], [X3DH] and [DR]**.

Application note: The key establishment and distribution as part of X3DH will describe the initial asynchronous setup of a shared secret between the TOE and another DCA user who are initiating secure messaging for the first time. The key distribution as part of Double Ratchet will describe continuous exchange of keys between a TOE and another DCA user engaging in secure messaging. These key exchanges are based on X448 Elliptic Curve Diffie-Hellman.

Application note: This SFR has been refined to specify that it involves key establishment rather than solely distribution. It includes the upload of keys to the DSS which acts as an intermediary between two DCA.

6.2.4 FCS_COP.1e– Cryptographic operation (EdDSA signatures)

FCS_COP.1.1 The TSF shall perform **signature generation and verification** in accordance with a specified cryptographic algorithm **EdDSA448** and cryptographic key sizes **456-bit** that meet the following: **[LIME] [X3DH], [RFC7748] and [RFC8032].**

Application note: This is the signature generation and verification operation as part of the LIMEv2 and X3DH protocols.

6.2.5 FCS_COP.1f – Cryptographic operation (X3DH key derivation)

FCS_COP.1.1 The TSF shall perform **X3DH key derivation** in accordance with a specified cryptographic algorithm **LIMEv2 HKDF based on SHA512** ~~**and cryptographic key sizes**~~ that meet the following: **[LIME], [RFC5869], [RFC2104] and ISO/IEC 10118-3:2018**.

Application note: This is the cryptographic operation at the end of the X3DH protocol which defines generation of a shared secret between two messaging parties. Note that while the HKDF function does accept an input a cryptographic key, it is simply a zeroized buffer as there are no shared secrets before this point of the messaging setup. Please see FCS_COP.1h for the HMAC algorithm.

6.2.6 FCS_COP.1g – Cryptographic operation (Double Ratchet key derivation)

FCS_COP.1.1 The TSF shall perform **Double Ratchet key derivation** in accordance with a specified cryptographic algorithm **LIMEv2 HKDF based on SHA512** and cryptographic key sizes **256-bit** that meet the following: **[LIME], [RFC5869], [RFC2104] and ISO/IEC 10118-3:2018**.

Application note: This SFR covers the cryptographic operation used for two purposes. One purpose is the chain key derivation and one is the message key derivation. As defined by the LIMEv2 protocol the input both in terms of the key and data differ depending on the key to be generated. The derived keys will provide the input for message encryption and the next Double Ratchet operation. For message encryption, the keys will be subject to Key Boosting before being utilized by the Dynamic Encryption algorithms. Please see FCS_COP.1h for the HMAC algorithm.

TLS channel encryption

These SFRs describe the cryptographic functionality for secure communication with the DSS. Note that this TLS channel is also used during calls and messaging, for instance during call initiation.

6.2.7 FCS_CKM.1a – Cryptographic key generation (TLS session key)

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **AES-256 in the Galois Counter Mode (GCM)** and specified cryptographic key sizes **256-bit (AES-256)** that meet the following: **[FIPS197] and [NIST SP 800-38D]**.

Application note: This SFR covers the generation of AES keys (sessions keys) for TLS connections.

6.2.8 FCS_CKM.2b – Cryptographic key distribution (TLS key establishment)

FCS_CKM.2.1 The TSF shall **perform** ~~**distribute**~~ cryptographic **key establishment** in accordance with a specified cryptographic key **establishment** ~~**distribution**~~ method **as defined in TLS 1.2 using Elliptic Curve Diffie-Hellman** that meets the following: **RFC5246 and RFC6347**

Application note: This SFR covers the key establishment and distribution which ensures the TOE is able to setup a trusted TLS channel and establish a common secret. TLS is used when connecting from the TOE to the DCS and DPS. This SFR has been refined to specify that it involves key establishment rather than solely distribution.

6.2.9 FCS_COP.1a – Cryptographic operation (AES data encryption and decryption)

FCS_COP.1.1 The TSF shall perform **encryption and decryption** in accordance with a specified cryptographic algorithm **AES-256 in GCM mode** and cryptographic key sizes **256-bit** that meet the following: **[FIPS197] and [NIST SP 800-38D]**.

Application note: This requirement addresses the data stream encryption and decryption of the TLS connections between the TOE and other parties. Note that this differs from Dynamic Encryption and is only used for the TLS channel with the DSS.

6.2.10 FCS_TLSC_EXT.1 – TLS client protocol

FCS_TLSC_EXT.1.1 The TSF shall implement **TLS 1.2 (RFC 5246)** and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites:

- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**

FCS_TLSC_EXT.1.2 The TSF shall verify that the presented identifier matches the reference identifier per RFC 6125 section 6.

FCS_TLSC_EXT.1.3 When establishing a trusted channel, by default the TSF shall not establish a trusted channel if the server certificate is invalid. The TSF shall also **not implement any administrator override mechanism.**

FCS_TLSC_EXT.1.4 The TSF shall **present the Supported Elliptic Curves Extension with the following NIST curves: secp384r1 and no other curves** in the Client Hello.

Application note: The unauthenticated TLS connection is only used for the provision connection of the TOE.

6.2.11 FCS_TLSC_EXT.2 – TLS client protocol with authentication

FCS_TLSC_EXT.2.1 The TSF shall implement **TLS 1.2 (RFC 5246)** and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites:

- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**

FCS_TLSC_EXT.2.2 The TSF shall verify that the presented identifier matches the reference identifier per RFC 6125 section 6.

FCS_TLSC_EXT.2.3 When establishing a trusted channel, by default the TSF shall not establish a trusted channel if the server certificate is invalid. The TSF shall also **not implement any administrator override mechanism.**

FCS_TLSC_EXT.2.4 The TSF shall **present the Supported Elliptic Curves Extension with the following NIST curves: secp384r1 and no other curves** in the Client Hello.

FCS_TLSC_EXT.2.5 The TSF shall support mutual authentication using X.509v3 certificates.

Application note: The ciphersuites used for the DPS webAPI connection and the DCS webAPI connection are the same. The authenticated TLS connection is for all TLS connections with exception of the provisioning TLS connection, i.e. for the initial configuration of the TOE.

General encryption

These SFRs describe the general cryptographic functionality of the TOE which are not used specifically for voice, video, messaging or DSS connections.

6.2.12 FCS_CKM.1b – Cryptographic key generation (RSA key generation)

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

- **RSA schemes using cryptographic key sizes of ~~2048-bit or greater~~ 3072-bit that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3**.

Application note: This SFR covers the generation of the private-public key pair for the DCA certificate. This requirement is a refinement of FCS_CKM.1.1 defined in [cPPND]. Before the client certificate expires, a new key pair is generated and the client certificate is renewed. After that, DCA has a new client certificate based on a new key pair.

6.2.13 FCS_CKM.1e – Cryptographic key generation (Storage key generation)

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **CTR_DRBG** and specified cryptographic key sizes **640-bit** that meet the following: **[NIST SP 800-90A]**

Application note: This SFR covers the generation of keys for the protection of storage. This involves two keys: one Storage Key (FCS_COP.1d) and one Device Key (FCS_COP.1j). The entropy source is Apple iOS RNG. Note that Dynamic Encryption requires an AES 256-bit key, a 128-bit seed for the S-Box generation, and two 128-bit whitening keys. This done by generating 640-bits and separating them into the necessary keys. The whitening keys can be zeroized, which they are in the application for the secure calls and secure messaging.

6.2.14 FCS_CKM.1d – Cryptographic key generation (Push key generation)

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **AES-256 in CFB mode** and specified cryptographic key sizes **256-bit (AES-256)** that meet the following: **[FIPS197] and [NIST SP 800-38A]**.

Application note: This SFR covers the generation of AES keys for push notification encryption.

6.2.15 FCS_CKM.2d – Cryptographic key distribution (Key distribution to DSS)

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method:

- **The TOE distributes an encrypted Storage Key, a Push Encryption Key and CSRs (RSA public key) to the DSS. All keys are transmitted via the trusted TLS channel to the DSS, which stores and associates it with the TOE user after the TOE has authenticated.**

that meets the following: **no standard.**

Application note: This SFR covers the distribution of cryptographic keys to the DSS. There are three different keys, but they are both distributed similarly over the TLS encrypted channel. The storage and push encryption keys are uploaded at the end of provisioning. For the X.509v3 client certificate only the public key is submitted along with relevant metadata as part of a CSR. The client generates the key-pair (FCS_CKM.1b), creates a CSR with the public key and sends the CSR to the DCM via the DCS or DPS. The DCM signs the CSR if permitted and returns a X.509v3 client certificate to the TOE.

6.2.16 FCS_CKM.4 – Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **zeroization** that meets the following: **no standard**.

Application note: Key destruction is performed of all symmetric keys that are generated by the TOE and used for data encryption and decryption.

6.2.17 FCS_COP.1b – Cryptographic operation (Digital signatures)

FCS_COP.1.1 The TSF shall perform **cryptographic signature services (generation and verification)** in accordance with a specified cryptographic algorithm **RSA Digital Signature Algorithm** and cryptographic key sizes:

- **3072-bit for generation**
- **3072-bit for verification**
- **4096-bit for verification**

that meet the following: **FIPS PUB 186-4 "Digital Signature Standard (DSS)" Section 5.5 using PKCS #1 v2.1 Signature Scheme RSASSA-PKCS1-v1.5 [FIPS186-4] [PKCS1v2.1].**

Application note: This requirement addresses the RSA signature generation and verification performed as part of the TLS server authentication performed by the TOE. This SFR applies also to the signature of the CSR sent by the TOE to the DCM via the DCS or DPS.

6.2.18 FCS_COP.1c – Cryptographic operation (Hash algorithm)

FCS_COP.1.1 The TSF shall perform **secure hash** in accordance with a specified cryptographic algorithms **SHA-384 and SHA-512** ~~**and cryptographic key sizes**~~ that meet the following: **ISO/IEC 10118-3:2018**.

Application note: The secure hash is used for both support integrity and key derivation with cryptographic operations of the TOE. It is used during TLS, DTLS-SRTP, LIMEv2, Key Boosting and during creation of X509 certificates.

Application Note: When this functionality is used for Key Boosting, it involves executing a number of consecutive hash operations; Key boosting utilizes 2^16 SHA-512 iterations.

6.2.19 FCS_COP.1d – Cryptographic operation (Dynamic Encryption and Decryption)

FCS_COP.1.1 The TSF shall perform **decryption and encryption** in accordance with a specified cryptographic algorithm **AES and modified AES in GCM mode for DCA** and cryptographic key sizes **256 and 128-bits** that meet the following: **[FIPS197] and [NIST SP 800-38D] and for the modified AES [Dynamic]**.

Application note: This requirement addresses both the AES data stream and modified (dynamic) encryption and decryption of the modified S-boxes as specified in the Dynamic Encryption. The operation is performed with a 256-bit key for AES, a 128 salt key for S-box generation, and two 128-bit whitening keys. Note that the whitening keys can be zeroized, as they are for secure calls and messaging.

Application note: Dynamic Encryption has the following uses by the TOE:

- Secure Call encryption using SRTP and SRTP-DTLS. This key is derived from the established common secret (FCS_CKM.2a) using Key Boosting. It involves executing a number of consecutive hash operations; Key boosting utilizes 2^16 SHA-512 iterations. (FCS_COP.1c).

- Secure Messaging using Double Ratchet. This key is derived from the derived Double Ratchet key (FCS_COP.1g) using Key Boosting. It involves executing a number of consecutive hash operations; Key boosting utilizes 2^16 SHA-512 iterations. (FCS_COP.1c).
- Dynamic Storage encryption using AES in GCM. This key is generated on the TOE (FCS_CKM.1e).

6.2.20 FCS_COP.1h – Cryptographic operation (Keyed-hash algorithm)

FCS_COP.1.1 The TSF shall perform **keyed-hash message authentication** in accordance with a specified cryptographic algorithm **HMAC-SHA-384 and HMAC-SHA-512** and cryptographic key sizes **256 and 384-bit** that meet the following: **ISO/IEC 10118-3:2018**.

Application note: This requirement addresses the HMAC operations as part of the TLS, DTLS-SRTP, X3DH and Double Ratchet protocols.

6.2.21 FCS_COP.1i – Cryptographic operation (Push decryption)

FCS_COP.1.1 The TSF shall perform **decryption** in accordance with a specified cryptographic algorithm **AES-256 in CFB mode** and cryptographic key sizes **256-bit** that meet the following: **[FIPS197] and [NIST SP 800-38A]**.

Application note: This requirement addresses the decryption of Push Notifications sent from the DSS to the TOE through the mobile OS vendor's systems. The key used is generated by the TOE and uploaded to the server during provisioning.

6.2.22 FCS_COP.1j – Cryptographic operation (Storage Key)

FCS_COP.1.1 The TSF shall perform **encryption and decryption** in accordance with a specified cryptographic algorithm **XOR** and cryptographic key sizes **640-bit** that meet the following: **no standard.**

Application note: This requirement specifies the encryption and decryption of the Storage Key. This encrypted key will then be sent to the DSS and is not stored locally by the TOE. When the TOE is started, it needs to authenticate to the DSS and import this key before being able to decrypt and read its storage. The Device Key is specific to this purpose in that it is only used to encrypt/decrypt this one Storage Key, and is only stored locally on the device.

General SFRs

These SFRs describe the general security functionality of the product, not including cryptography.

6.2.23 FDP_IFC.2 – Complete information flow control

FDP_IFC.2.1 The TSF shall enforce the **GROUP information flow control SFP** on **TOE instances and voice, video or message data** and all operations that cause that information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2 The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

Application note: The calls and messages are restricted to the phone book entries held by the TOE. It is not possible for users to add, delete or modify these entries. The phone book is centrally managed.

6.2.24 FDP_IFF.1 – Simple security attributes

FDP_IFF.1.1 The TSF shall enforce the **GROUP information flow control SFP** based on the following types of subject and information security attributes: **TOE instances (user handsets) and user name**.

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: secure calls and secure messaging can only be established from a TOE to another instance of the DCA when the name of the user of the second DCA instance (callee) is in the phone book of the caller.

FDP_IFF.1.3 The TSF shall enforce **no additional rules**.

FDP_IFF.1.4 The TSF shall explicitly authorise an information flow based on the following rules: **none**.

FDP_IFF.1.5 The TSF shall explicitly deny an information flow based on the following rules: **none**.

Application note: A user's phone book only contains the names of the other users he/she is allowed to call or message to. Please note a pre-condition for this is that all these users have been issued a permanent certificate by the DCM. If the receiver does not have the TOE user in the phone book (there are cases where a small group of users can contact the users in a bigger group, but not everyone in the bigger group are allowed to contact the users in the small group), the TOE user's display name is shown on the incoming call screen. The TOE user's display name is provided as metadata in call setup message, i.e. the server system defines what is displayed on the incoming call screen. For messages, the TOE user's identity is specified during the initial messaging setup and verified by the identity keys.

6.2.25 FDP_ITC.1 – Import of user data without security attributes

FDP_ITC.1.1 The TSF shall enforce the **Encryption Key SFP** when importing user data, controlled under the SFP, from outside of the TOE.

Application note: The TOE does not store the Storage Key locally on the device. This key is encrypted by a Device Key, which is stored locally. Therefore, when the TOE needs to access its stored encrypted contents, it first needs to authenticate and retrieve the Storage Key from the DSS. The keys are received over a mutually authenticated channel, as specified in FCS_TLSC_EXT.2. There is no additional flow or access control policy imposed on the object.

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

Application note: This is the only key imported by the TOE and it in of itself has no attributes.

FDP_ITC.1.3 The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **the key must not be stored persistently by the TOE.**

Application note: The key must not be stored locally by the TOE and should only remain in RAM for as long as the TOE is active. When the TOE is restarted, a successful authentication with the server is necessary to retrieve the key.

6.2.26 FMT_MTD.1 – Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to **modify** the **network settings and phone book** to **the remote DCS connection**.

Application note: The TOE management functions are performed in the TOE environment by the administrator using the DCC. The settings are then pushed to the TOE from the DCS whenever a DCS connection is made. The TOE user is not allowed to modify the network settings and phone book on the TOE.

6.2.27 FMT_SMF.1 – Specification of management functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:

- **download network settings and phone book**
- **replace the existing network settings and phone book with the downloaded versions**

Application note: The network settings and phone book on the TOE are kept in sync with server system's data whenever the TOE is running and registered on the DCS. Note: These management functions are not performed by any user but are performed automatically by the TOE.

6.2.28 FTP_ITC.1a – Inter-TSF trusted channel (TLS)

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

FTP_ITC.1.2 The TSF shall permit the **TSF** to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for **connecting to DSS**.

Application note: This is the trusted channel (TLS) between the TOE and the DCS and DPS. Note that this is only used for the TOE and not for access to any other systems or corporate resources (such as Intranet or email). The cryptography is described in FCS_TLSC_EXT.1 and in FCS_TLSC_EXT.2.

Application note: The TLS connection to the provisioning web server is not authenticated by the server but the provisioning data can be established only once and for a limited time after the link has been provided and only by the TOE user that knows the unique link.

6.2.29 FTP_ITC.1b – Inter-TSF trusted channel (VoIP)

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

FTP_ITC.1.2 The TSF shall permit **the TSF <u>or another instantiation of the TOE</u>** to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for **secure voice and video**.

Application note: This channel is the end-to-end encrypted channel between the TOE and another instantiation of the TOE that is used for encrypted voice and video (VoIP).

6.3 Security functional requirements rationale

6.3.1 Coverage

The following table provides a mapping of SFR to the security objectives, showing that each security functional requirement addresses at least one security objective and that all security objectives are addressed by one or more SFRs.

	O.CALLERID	O.GROUP	O.TRAFFIC	O.MESSAGES	O.CHANNEL	O.PHONEBOOK	O.PRIVATEKEY	O.FORWARD	O.MANAGE	O.STORAGE	O.TUNNEL	O.PUSH
FCS_CKM.1a (TLS session AES)					X							
FCS_CKM.1b (RSA keypair)			X		X		X					
FCS_CKM.1c (LIME EdDSA)				X								
FCS_CKM.1d (AES CFB key)												X
FCS_CKM.1e (Storage encryption)										X		
FCS_CKM.2a (DTLS-SRTP key establishment)			X									
FCS_CKM.2b (TLS key establishment)					X							
FCS_CKM.2c (LIME key establishment)				X								
FCS_CKM.2d (DSS key distribution)					X					X		X
FCS_CKM.4			X	X	X			X				
FCS_COP.1a (TLS AES)					X							
FCS_COP.1b (Digital signature)			X		X							
FCS_COP.1c (Hash)			X	X	X							
FCS_COP.1d (Dynamic AES)			X	X						X		
FCS_COP.1e (EdDSA)				X								
FCS_COP.1f (X3DH)				X								
FCS_COP.1g (Double Ratchet)				X								
FCS_COP.1h (Keyed-hash)			X	X	X							
FCS_COP.1i (AES Push decryption)												X
FCS_COP.1j (Storage key encryption)										X		

	O.CALLERID	O.GROUP	O.TRAFFIC	O.MESSAGES	O.CHANNEL	O.PHONEBOOK	O.PRIVATEKEY	O.FORWARD	O.MANAGE	O.STORAGE	O.TUNNEL	O.PUSH
FCS_TLSC_EXT.1									X			
FCS_TLSC_EXT.2			X	X	X						X	
FDP_IFC.2	X	X										
FDP_IFF.1	X	X										
FDP_ITC.1										X		
FMT_MTD.1						X			X			
FMT_SMF.1						X			X			
FTP_ITC.1a (DSS)			X	X	X					X	X	X
FTP_ITC.1b (VoIP)	X		X									

6.3.2 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	Security objectives
O.CALLERID	The objective: • The TOE must ensure that the end point of a secure call or messaging connection is unique and that the caller display name associated with the caller identity is correctly shown to the TOE user making or receiving the secure call or message. is met by: • FDP_IFC.2 and FDP_IFF.1 ensuring that information flow is only possible between parties represented in the caller's phone book. • FTP_ITC.1b ensuring that there is a mutually authenticated trusted channel between the two parties
O.GROUP	The objective: • The TOE must ensure that secure calls and messaging is restricted to users within the TOE phone book of the calling party. is met by: • FDP_IFC.2 and FDP_IFF.1 ensuring that information flow is only possible between parties represented in the caller's phone book.
O.TRAFFIC	The objective: • The TOE must ensure that secure calls are protected against disclosure and modification. is met by: • FTP_ITC.1b ensures that there is a trusted path for secure voice and video between the TOE and a different instance of the DCA. • FCS_CKM.1b and FCS_CKM.2a ensures secure and authenticated cryptographic key establishment via the DTLS-SRTP protocol. • FCS_COP.1b, FCS_COP.1c, FCS_COP.1d and FCS_COP.1h ensure the cryptographic functionality to achieve confidentiality, integrity and authenticity.

Security Objective	Security objectives
	• FTP_ITC.1a ensures that there is a trusted path between the TOE and the DSS, which is used to initiate the calls. • FCS_TLSC_EXT.2 ensures that this path is cryptographically protected by TLS. • Key destruction is performed by FCS_CKM.4
O.MESSAGES	The objective: • The TOE must ensure that secure messages are protected against disclosure and modification. is met by: • FCS_CKM.1c ensures cryptographic key generation of the necessary identity keys and pre-keys for the protocol. • FCS_CKM.2c ensures secure key establishment and distribution. • FCS_COP.1c, FCS_COP.1f, FCS_COP.1g and FCS_COP.1h ensure the necessary key derivation functionality providing forward secrecy. • FCS_COP.1e ensure signature generation and validation. • FCS_COP.1d ensures confidentiality via encryption and decryption of messages. • FTP_ITC.1a ensures that there is a trusted path between the TOE and the DSS, which is used to upload the messages and keys. • FCS_TLSC_EXT.2 ensures that this path is cryptographically protected by TLS. • Key destruction is performed by FCS_CKM.4
O.CHANNEL	The objective: • The TOE must ensure that there is a trusted path between the TOE and the server system ensuring authenticity, integrity and confidentiality and integrity of any TSF or user data transmitted between the TOE and the server system, such as phone book updates and SIP connections made when establishing secure calls. is met by: • FTP_ITC.1a that ensures that there is a trusted path between the TOE and the DSS. • FCS_CKM.2b ensures ECDH key establishment. • Key generation is done by FCS_CKM.1a and FCS_CKM.1b. • Key distribution of the public key (CSR) for mutual authentication is done by FCS_CKM.2d. • The TLS 1.2 protocol implemented by FCS_TLSC_EXT.2. • FCS_COP.1a, FCS_COP.1b, FCS_COP.1c and FCS_COP.1h ensure the cryptographic functionality to achieve confidentiality, integrity and authenticity. • Key destruction is performed by FCS_CKM.4.
O.PHONEBOOK	The objective: • The TOE must ensure that the phone book cannot be changed locally. is met by: • FMT_SMF.1 ensures that the network settings and the phone book can be managed • FMT_MTD.1 ensures that this is restricted to the remote network connection to the DCS
O.PRIVATEKEY	The objective:

Security Objective	Security objectives
	• The TOE must be able to generate its own key pair. is met by: • FCS_CKM.1b ensures that the private/public key pair for the TOE is generated by the TOE itself.
O.FORWARD	The objective: • The TOE must ensure that an unauthorized user that obtains a handset cannot decrypt previously transmitted traffic (voice, video or messages) that has been encrypted using the obtained handset. Any encryption keys used for the transmitted traffic must be erased. is met by: • FCS_CKM.4 ensures that keys used for encryption and decryption of secure voice, video and messages are destroyed after a secure call is completed or after a secure message is decrypted.
O.MANAGE	The objective: • The TOE must support provisioning and ensure that settings and phone book can be updated whenever the TOE is running and registered on the DCS. is met by: • FMT_SMF.1 ensures that the network settings and the phone book can be managed. • FMT_MTD.1 ensures that this is restricted to remote management. • FCS_TLSC_EXT.1 ensures that the initial credentials are provisioned to the TOE by a trusted server system and the credentials are protected by TLS during transmission.
O.STORAGE	The objective: • The TOE must use cryptography to protect the confidentiality of DCA data stored on the device. The data must only be accessible if the user can authenticate to the DSS. is met by: • FCS_CKM.1e ensures key generation. • FCS_CKM.2d ensures that the encrypted Storage Key can be distributed to the trusted DSS. • FCS_COP.1d ensures that the stored data and message file attachments can be encrypted and decrypted. • FCS_COP.1j ensures that the Storage Key can be encrypted and decrypted. • FDP_ITC.1 ensures that the encrypted Storage Key can be retrieved from the trusted DSS. • FTP_ITC.1a ensure that the key is transmitted over the encrypted channel.
O.TUNNEL	The objective: • The TOE must support tunnelling of voice or video communication over TCP by tunnelling the connection over TLS 1.2. is met by: protocol which is used by the tunnel. • FCS_TLSC_EXT.2 enables the TCP tunnel via TLS 1.2. • FTP_ITC.1a ensures that the encrypted tunnel can be established.
O.PUSH	The objective:

Security Objective	Security objectives
	• The TOE must be able to decrypt Push Notifications that are sent via the mobile operating system vendor's systems. is met by: • FCS_CKM.1d ensures key generation. • FCS_CKM.2d ensures key distribution to the DSS. • FCS_COP.1i ensures decryption of push notifications from the DSS. • FTP_ITC.1a ensures that the encrypted tunnel can be established.

6.3.3 Dependency analysis between security functional components

The following table shows the dependencies of the SFRs and shows how these dependencies have been resolved.

SFR	Dependencies	Resolved?
FCS_CKM.1a (TLS session AES)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1a Yes, by FCS_CKM.4
FCS_CKM.1b (RSA keypair)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_CKM.2b Yes, by FCS_CKM.4
FCS_CKM.1c (LIME EdDSA)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_CKM.2c Yes, by FCS_CKM.4
FCS_CKM.1d (AES CFB key)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1i Yes, by FCS_CKM.4
FCS_CKM.1e (Storage key)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1d. Yes, by FCS_CKM.4
FCS_CKM.2a (DTLS-SRTP key establishment)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1b Yes, by FCS_CKM.4
FCS_CKM.2b (TLS key establishment)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1b Yes, by FCS_CKM.4
FCS_CKM.2c (LIME key establishment)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1c Yes, by FCS_CKM.4
FCS_CKM.2d (DSS key distribution)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1b, FCS_CKM.1d and FCS_CKM.1e Yes, by FCS_CKM.4
FCS_CKM.4	No dependencies	--

SFR	Dependencies	Resolved?
FCS_COP.1a (TLS AES)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1a Yes, FCS_CKM.4
FCS_COP.1b (Digital signature)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1b No, no key destruction is needed because the private key is kept in the TOE for use as long as the corresponding certificate is valid.
FCS_COP.1c (Hash)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	No, no key is needed for SHA-384 or SHA-512. No, no key destruction needed
FCS_COP.1d (Dynamic AES)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1e. These keys are also derived from key establishment methods FCS_CKM.2a and FCS_CKM.2c. Yes, by FCS_CKM.4
FCS_COP.1e (EdDSA)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1c Yes, by FCS_CKM.4
FCS_COP.1f (X3DH)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1c. Note that keys are additionally derived and established using FCS_CKM.2c Yes, by FCS_CKM.4
FCS_COP.1g (Double Ratchet)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1c. Note that keys are also derived from FCS_COP.1f and FCS_COP.1g, and shared secrets are established using FCS_CKM.2c. Yes, by FCS_CKM.4
FCS_COP.1h (Keyed-hash)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	No. The key generation for HMAC used within TLS are generated via establishing the common secret in FCS_CKM.2b. The key generated for HMAC within Double Ratchet is derived by the HKDF specified by FCS_COP.1g or FCS_COP.1h Yes, by FCS_CKM.4
FCS_COP.1i (AES Push decryption)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1d Yes, by FCS_CKM.4
FCS_COP.1j (Storage key encryption)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FDP_ITC.1. Note that the first instance of this key is generated via FCS_CKM.1e Yes, by FCS_CKM.4
FCS_TLSC_EXT.1	FCS_CKM.1 FCS_CKM.2 FCS_COP.1/ DataEncryption	Yes, by FCS_CKM.1b Yes, by FCS_CKM.2b Yes, by FCS_COP.1a

SFR	Dependencies	Resolved?
	FCS_COP.1/SigGen FCS_COP.1/Hash FCS_COP.1/ KeyedHash FCS_RBG_EXT.1	Yes, by FCS_COP.1b Yes, by FCS_COP.1c Yes, by FCS_COP.1h No, but addressed by OE.KEYS
FCS_TLSC_EXT.2	FCS_CKM.1 FCS_CKM.2 FCS_COP.1/ DataEncryption FCS_COP.1/SigGen FCS_COP.1/Hash FCS_COP.1/ KeyedHash FCS_RBG_EXT.1	Yes, by FCS_CKM.1b Yes, by FCS_CKM.2b Yes, by FCS_COP.1a Yes, by FCS_COP.1b Yes, by FCS_COP.1c Yes, by FCS_COP.1h No, but addressed by OE.KEYS
FDP_IFC.2	FDP_IFF.1	Yes, by FDP_IFF.1
FDP_IFF.1	FDP_IFC.1 FMT_MSA.3	Yes, by FDP_IFC.2 No, attributes are static and assigned as part of the FMT_MTD.1
FDP_ITC.1	[FDP_ACC.1 or FDP_IFC.1] FMT_MSA.3	No, no dedicated Access Control or Information Flow Control policy is used since this import takes place over a mutually identified and authenticated TLS session initiated by the trusted DSS. Only the Storage Key is imported and no additional controls are imposed. No, the imported value is an encrypted Storage Key, which is generated by FCS_CKM.1e and then encrypted by the TOE via FCS_COP.1j
FMT_MTD.1	FMT_SMR.1 FMT_SMF.1	No, management functions are performed by the backend connection to the DCS and the TOE. Yes, by FMT_SMF.1
FMT_SMF.1	No dependencies	–
FTP_ITC.1a	No dependencies	–
FTP_ITC.1b	No dependencies	–

6.4 Security assurance requirements

The security assurance requirements of this Security are those defined for the assurance level EAL4 augmented with ALC_FLR.2.

Assurance class	Assurance components
ADV: Development	ADV_ARC.1 Security architecture description
	ADV_FSP.4 Complete functional specification
	ADV_IMP.1 Implementation representation of the TSF
	ADV_TDS.3 Basic modular design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures

Assurance class	Assurance components
ALC: Life-cycle support	ALC_CMC.4 Production support, acceptance procedures and automation
	ALC_CMS.4 Problem tracking CM coverage
	ALC_DEL.1 Delivery procedures
	ALC_DVS.1 Identification of security measures
	ALC_FLR.2 Flaw reporting procedures (augmentation)
	ALC_LCD.1 Developer defined life-cycle model
	ALC_TAT.1 Well-defined development tools
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.2 Analysis of coverage
	ATE_DPT.1 Testing: basic design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing – sample
AVA: Vulnerability assessment	AVA_VAN.3 Focused vulnerability analysis

6.5 Security assurance requirements rationale

Dependencies within the EAL package selected (EAL4) for the security assurance requirements have been considered by the authors of CC Part 3 and are not analysed here again. The augmentation by flaw remediation, ALC_FLR.2, has no dependencies on other requirements. The security functional requirements in this Security Target do not introduce dependencies on any security assurance requirement; neither do the security assurance requirements in this Security Target introduce dependencies on any security functional requirement.

The assurance level EAL4 augmented with ALC_FLR.2 has been chosen since EAL4 is the lowest assurance package which includes source-code analysis. The source code analysis is necessary to assess the implementation quality and ensure that the TOE does not contain any malicious code.

EAL4 is augmented by ALC_FLR.2 as during operations new vulnerabilities may be discovered, either through developer actions (e.g., developer testing) or those discovered by others. It requires the developer to have procedures addressing these vulnerabilities. The process used by the developer corrects any discovered vulnerabilities and performs an analysis to ensure that no new vulnerabilities are created while fixing the discovered ones.

7 TOE Summary Specification

The TOE summary specification identifies the security functions that the TOE implements to meet the requirements defined in chapter 6 to the security target.

The table below shows which SFRs are satisfied by each of the TSFs.

TSF	SFRs met by the TSF
SF.PROVISIONING	FTP_ITC.1a FMT_MTD.1
SF.MANAGEMENT	FCS_CKM.1b FCS_CKM.2d (for certificate signing request) FMT_MTD.1 FMT_SMF.1
SF.CALLS	FCS_CKM.2a (DTLS-SRTP) FCS_COP.1d (AES + modified AES) FCS_COP.1b (Signatures) FCS_COP.1c (Hashing, key boosting) FCS_COP.1h (HMAC) FCS_CKM.4 FDP_IFC.2 (GROUP) FDP_IFF.1 (GROUP) FTP_ITC.1b (Call traffic) FTP_ITC.1a (Call initialisation)
SF.MESSAGES	FCS_CKM.1c FCS_CKM.2c (X3DH Key establishment) FCS_CKM.4 FCS_COP.1d (AES + modified AES) FCS_COP.1e (Signatures) FCS_COP.1c (Hashing, key boosting) FCS_COP.1f (X3DH key derivation) FCS_COP.1g (Double Ratchet key derivation) FCS_COP.1h (Used by key derivation) FDP_IFC.2 (GROUP) FDP_IFF.1 (GROUP) FTP_ITC.1a (Message routing)
SF.STORAGE	FCS_CKM.1e FCS_CKM.2d FCS_COP.1d (Data encryption) FCS_COP.1j (Key encryption/decryption) FDP_ITC.1 (Key import) FTP_ITC.1a
SF.CHANNEL	FCS_COP.1a (AES for TLS) FCS_COP.1b (Signatures) FCS_COP.1c (Hashing) FCS_COP.1h (HMAC) FCS_CKM.1a FCS_CKM.1b FCS_CKM.2b (Key establishment) FCS_TLSC_EXT.1 FCS_TLSC_EXT.2

TSF	SFRs met by the TSF
	FTP_ITC.1a
SF.PUSH	FCS_CKM.1d FCS_CKM.2d FCS_COP.1i FTP_ITC.1a
SF.TUNNEL	FCS_TLSC_EXT.2 FTP_ITC.1a

7.1 SF.PROVISIONING – Secure initialisation

The provisioning process consist of two steps, the installation of the DCA and the provisioning of the user credentials. The installation of the DCA is not part of the TSF, but the provisioning of the user credentials to the TOE is, which is described here.

Provisioning starts by the Dencrypt administrator by adding the user to the DSS. Afterwards, the administrator provides an invitation link to the user. This link can also be encoded as a QR-code. This link must be provided in a secure way to the user, i.e. the link is not disclosed during transmission to the TOE user. The invitation link might be mailed to the TOE if the mail transmission between mail server and handset's mail client is encrypted and the mail server is controlled by the organisation of the TOE's user. Note, SMS does not meet the requirement of non-disclosure because the mobile operator that transmits the SMS has access to the its content, the invitation link.

The invitation link points to the web server of the DPS. This is all done in the TOE environment.

The user shall tap the invitation link or scan the QR code which starts the TOE. The TOE parses the link, fetches the provisioning data from the DPS and installs the provisioning data. This also involves the generation of the first certificate and key pair on the TOE, where a CSR is submitted to the DSS. The provisioning data are deleted on the DPS, i.e. the HTTPS link can be used only once. Additionally, the link is only valid for a limited time after the link has been provided.

The network settings and phone book of the DCA (TOE) are updated as described in chapter “Managing settings and phone book”. After that the TOE is fully setup and operational.

7.2 SF.MANAGEMENT – Update of TOE settings, phone book and certificate

When a SIP registration is successful, the client will poll the DSS for updated settings. This allows the TOE to keep its local settings in sync with the server system's settings. Whenever the TOE is running and registered on the DCS, the TOE downloads network settings as soon as the checksum of its local settings differs from the settings checksum advertised by the DCS. The same mechanism applies to keep the phone book up-to-date.

The security settings that are updated are the:

- Network settings
- App settings
- Phone book

Note: These security settings are managed only by remote administrators that are identified and authenticated by the TOE environment of the DSS The TOE simply downloads the settings when new versions become available. Although the TOE user is the single individual that is authorized to use the TOE, the user cannot change any phone book entries or make calls from a dial-pad:

- This minimizes the risk of a phishing attack.
- This keeps the administrator in full control of the phone book content.
- This adds a layer of security because only people that have been given permission can call you using the DCA.

Besides, the mobile's built-in phone book is kept completely separate from the DCA's phone book. Hence, removing DCA from the mobile phone does not leave "traces" in the built-in phone book and deploying a new mobile simply means to provision the user on a new phone.

Apart from phone book updates, the TOE will generate by itself an RSA 3072-bit private-public key pair and send its public key with a certificate signing request (CSR) to the DCM server which signs it and delivers the client certificate. This is to ensure that the private key never must leave the TOE.

The client creates a new private key and CSR if the client certificate expires soon (e.g. within the next 12 months). This CSR is submitted to the DCM via the DCS, since the DPS is only used during provisioning. This TLS session is mutually authenticated, so if the client certificate has expired it would not be possible and require new provisioning.

7.3 SF.CALLS – Secure voice and video

The secure communication channel between two handsets, the TOE and other instance of the TOE goes as followed: Dencrypt's secure calls extends a Voice/Video over IP (VoIP) system by a patented Dynamic Encryption for voice and video data. Dencrypt's VoIP system employs the Session Initiated Protocol (SIP), Secure Real-time Transmission Protocol (SRTP) (RFC 3711) and DTLS-SRTP (RFC 5763) standard components that are partly modified to integrate dynamic encryption.

The core elements of the SIP VoIP system are the SIP clients and the SIP server. The SIP server is the signalling center of the communication system, e.g. contacting the SIP client for call, giving the start signal for the waiting tone of the calling party and terminating the call as soon as one side is releasing the call. Hence, the SIP session is named "call session". SIP transmissions are protected by mutually authenticated TLS. The SIP VoIP system is designed for the IP network, i.e. Dencrypt Communication Solutions can be deployed where an IP network connection exits but is limited to Dencrypt Communication Solution users.

The SRTP components are implemented in the SIP clients and secure the audio data stream. SRTP is the secure variant of RTP. (S)RTP sessions are named media or content sessions. The DTLS-SRTP protocol is used to establish the keys that are needed by Dynamic Encryption of the call. The following figure displays the relationship between different keys for Dynamic Encryption in DCA. Note that the implementations of secure call and messaging are different.

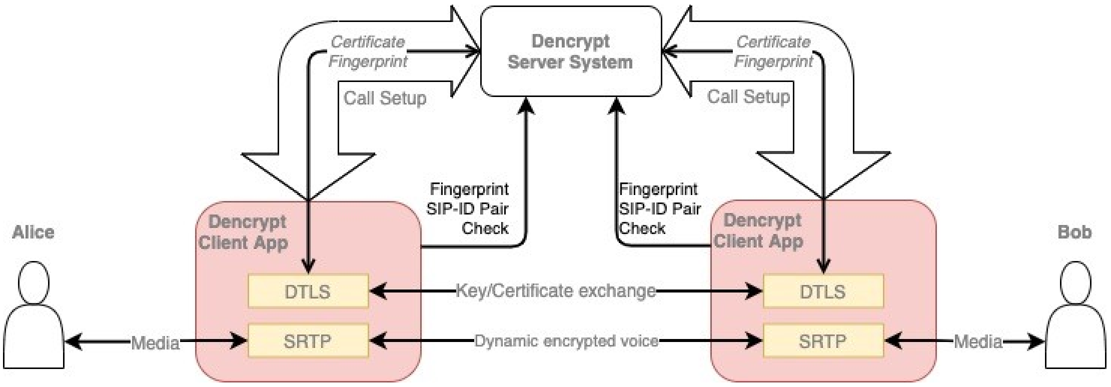

Illustration 5: Secure call cryptography overview

For secure calls, DTLS-SRTP uses Elliptic Curve Diffie-Hellman key exchange to establish a common secret and authenticate the call. Additionally, both DCAs exchange their client certificates over the peer-to-peer connection. The fingerprints of the certificates are sent over the call setup channel through the DSS. To mitigate a Man-in-the-middle (MitM) attack, the fingerprint calculated from the exchanged certificate has to match with the fingerprint signalled through the DSS. The parties validate the calculated finger print with the certificate fingerprint received via SIP. The DSS provides verification that the fingerprint matches the other party's SIP ID. The common secret can now be used to derive a 256-bit key and 128-bit salt which is fed into SRTP's key derivation function (KDF). As specified in RFC3711, the KDF generates a message authentication session key, a block cipher session key and a salt session key. To complete the Dynamic Encryption input Key Boosting is used to create a common Dynamic Encryption key which is used for both AES and the S-box seed. Once the SRTP session ends all the keys are destroyed (overwritten with zeros). But all the content, i.e. SRTP stream for audio and text data for messaging, are dynamically encrypted. Details of the Dynamic Encryption are given in section 7.3.1.

Secure calls and messages can only be initiated to users within the caller's phone book. Note that the DCA user can receive calls from users who are not listen in their phone book. The phone book is centrally managed and the TOE user is not able to add, delete or modify it. If the call receiver does not have the caller in the phone book (there are cases where a small group of users can call the users in a bigger group, but not everyone in the bigger group are allowed to call the users in the small group), the caller's display name is shown on the incoming call screen. This display name is defined by the DSS and is provided as metadata in the call setup message.

7.3.1 Dynamic Encryption

Dencrypt's patented Dynamic Encryption was invented by Lars Ramkilde Knudsen. The patent is identified by its patent number WO2013060876.

Key elements of the patent:

1. The decryption method is not pre-defined but will be determined during the communication establishment.
2. The sender of the encrypted data determines how the receiver shall decrypt the received encrypted data.

Dynamic encryption relies on AES encryption for security, but adds a layer of obfuscation to AES.

Dencrypt's Implementation Concept

The patent itself does not define how to setup the decryption method on the receiver. Dencrypt's implementation of Dynamic Encryption adds an additional layer around the standard AES cipher. The additional layer and the standard AES cipher together form a new cipher: The Dynamic Encryption cipher. The additional layer is determined by the whitening keys and the substitution-box (S-box) where the S-box is determined by a given S-Box seed that is given before the encryption session. The S-Box generator is responsible to return a secure S-Box based on a given seed. Additional xor-ing by two different whitening keys is applied before and after AES encryption and S-Box substitution to strengthen the Dynamic Encryption cipher. Note that the whitening keys can be zeroised, which is the case with Secure Calls and Secure Messages.

For decryption, the inverse S-Box is calculated and the algorithm is executed backwards.

Mode of Operation

Dynamic encryption uses AES which is a block cipher that handle blocks of 128-bits. To ensure confidentiality, the blocks are interconnected. Dynamic Encryption in the TOE uses the Galois/Counter Mode (GCM).

7.3.2 Key Boosting

Dencrypt’s patent Key Boosting was invented by Lars Ramkilde Knudsen. The patent is identified by the application number: EP appl. no.: 19197344.5

The key element of Key Boosting is that it is a method to generate additional secure bits from a symmetric key. The additional bits are generated using a one-way function. The implementation can vary based upon the number of generated secure bits and the one-way function.

The secure bits are a combination of the security of the initial key and the computational effort of computing 2^n rounds of the one-way function SHA512. The patent assumes that the computational complexity of guessing a bit is the same as 2 rounds of the one-way function. The TOE performs the Key Boosting by iterating the one-way function 2^16 times.

7.4 SF.MESSAGES – Secure messaging

The Secure Messaging is provided by the LIMEv2 protocol, which in turn is based on the Signal Protocol. Additionally the message encryption has been modified via Dencrypt’s Dynamic Encryption and Key Boosting, which are the same implementation as for Secure Calls 7.3.1 and 7.3.2. Secure messaging is implemented very differently from the secure calls. Messaging is asynchronous unlike secure calls and each message has to be encrypted individually. The LIMEv2 protocol also provides forward secrecy.

The DSS will route and store messages and public keys between the clients. The DCA primarily generates asymmetric keypairs and uploads its public keys to the DSS. These include both identity keys and Diffie-Hellman keys to derive the message keys. Each message will require a new message key.

X3DH establishes the initial shared secret. The multiple DH operations require public DH keys of the receiver which are downloaded from the DSS. Thus, all users maintain a pool of public keys for all their devices on the DSS. X3DH is only required to start a new encryption session, i.e. a new encrypt chat conversation. The algorithm is omitted for the later messages in the same conversation.

Double Ratchet computes the symmetric key for the Dynamic Encryption of the messages. The algorithm ratchets the symmetric for each message to ensure Perfect Forward Secrecy (PFS). If a message is received, the Double Ratchets performs a Diffie-Hellman which features break-in recovery. Note the difference between the DH ratchet and the symmetric ratchet.

The 256-bits exchanged message keys are key boosted utilizing 2^16 SHA-512 iterations to get a 384-bit Dynamic Encryption key. The messages are encrypted by Dynamic Encryption where the whitening keys are set to 0. The deployed mode of operation is GCM. After a message is received and decrypted, the related keys are erased.

7.5 SF.STORAGE – Data-at-rest protection

The TOE dynamically encrypts the databases that store the chat history and the LIMEv2 keys. The same Dynamic Encryption key decrypts both databases and the key are constant but unique for each TOE. The Dynamic Encryption principle is applied because each device has a unique encryption algorithm.

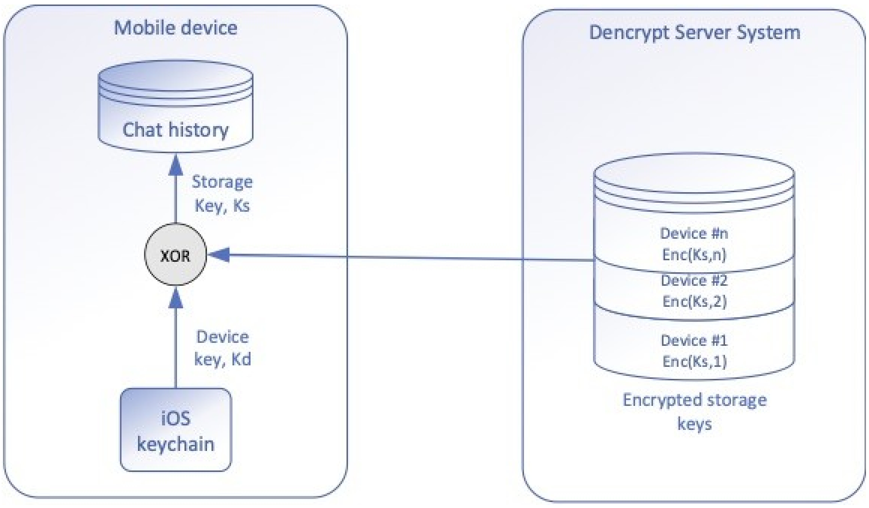

Illustration 6: Storage encryption overview

At the end of provisioning the TOE generates two 640-bit keys. One (Storage Key) will be used for Dynamic Encryption of the storage, while one (Device Key) will be used as a Key Encryption Key using XOR. The encrypted Storage Key is submitted to the DSS and will not be stored by the TOE, it is downloaded and only kept in RAM. After a message is received and decrypted using the LIMEv2 protocol it is encrypted via storage encryption before being stored on the device. The cleartext data (messages and attachments) will only be stored in RAM.

The Device Key will be stored in the Secure Enclave of the iPhone, and will only be used to decrypt the Storage Key.

The approach ensures that neither the device nor the server system becomes a point-of-attack. In case of a lost or stolen device, the TLS certificate is revoked and the device cannot connect to the server system to fetch the encrypted Storage Key.

The approach also implies that the chat history is not available for the end-user in offline mode. The administrator can disable the upload of the encrypted storage key but it is stored in DCA's HSM instead. However, offline access to the encrypted DB comes with the price of reduced security.

7.6 SF.CHANNEL – Secure communication channel (TLS)

The TOE can establish a secure channel between the TOE and DSS components. All TLS connections are initiated by the TOE.

The secure SIP connection between the TOE and the SIP server on the DCS uses mutually authenticated TLS 1.2, with RSA signature, AES 256-bit encryption and SHA-384 hashing.

The HTTPS connection to the web server (webAPI) on the Dencrypt Provisioning Server (DPS) also uses TLS 1.2, with RSA signature, AES 256-bit encryption and SHA-384 hashing. However, in this case the TOE is not authenticated to the DPS (i.e. only the server side of the TLS channel is authenticated). This is only performed once during the provisioning and the link is only valid for a limited time after it has been provided.

For TLS 1.2 mutual authentication, the TOE has a 3072-bit RSA key pair while each server component has a 4096-bit RSA key pair. So, the TOE shall verify server signature by using the server system's 4096-bit RSA public key.

7.7 SF.PUSH – Encrypted push notifications

Apple requires the TOE to indicate to the user immediate feedback for incoming mobile push. This leaves TOE no time to connect to DSS to get hold of the incoming caller ID. The SIP caller ID is placed encrypted in the mobile push. This ensures that only the intended DCA recipient can read the caller ID, even though the push message is sent via the system of a different vendor.

The TOE generates and submits an AES push encryption key at the end of provisioning. The key is also stored by the TOE and used to decrypt incoming push notifications. AES CFB is used instead of Dynamic Encryption since DSS has no Dynamic Encryption capabilities. There is no integrity protection since the push notification is sent through Apple via TLS.

7.8 SF.TUNNEL – TCP tunnelling of secure calls

The TOE also supports tunnelling of voice and video traffic over TCP and TLS 1.2. Voice and video are normally sent over UDP via SRTP, however, this tunnelling service will allow the TOE to instead use TCP and TLS 1.2 to tunnel the call traffic. The DSS provides a tunnel server to which the TOE can connect to enable this. Whether the tunnel is enabled or not by default depends on the settings as specified on the server, but this setting can be changed on the TOE.

While the tunnel provides the TLS encryption, this is not security relevant since the tunnelled traffic is still protected by SRTP.

7.9 Cryptographic functions and parameters

This section summarizes the cryptographic mechanisms and primitives and parameters used by the TSFs previously described.

Dynamic Encryption	Used by **SF.CALLS, SF.MESSAGES and SF.STORAGE** Uses AES-256 128-bit seed for S-Box generation, i.e. 128-bit for algorithm selection 2 x 128-bit Whitening keys (xor'ing). Note that these can be zeroized. Used by SRTP, Double Ratchet and Storage encryption.
SRTP (RFC 3711)	Used by **SF.CALLS** Modified by Dencrypt to support Dynamic Encryption cipher suite: • SRTP_DYNCIPHER_KB16_GCM_256 (which is based on AEAD_AES_256_GCM specified in RFC7714) Key boosting with 2^16 SHA-512 iterations, whitening keys are zeroised
DTLS-SRTP (RFC 5763, RFC5764)	Used by **SF.CALLS** DTLS 1.2 Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 Elliptic curve: secp384r1 Certificate fingerprint: SHA-512
X3DH	Used by **SF.MESSAGES** Elliptic Curve Diffie-Hellman with curve X448 HKDF: SHA512 (RFC5869) without any key Signatures: EdDSA448
Double Ratchet	Used by **SF.MESSAGES** Elliptic Curve Diffie-Hellman with curve X448 Modified by Dencrypt to support Dynamic Encryption cipher suite: • SRTP_DYNCIPHER_KB16_GCM_256 (which is based on AEAD_AES_256_GCM specified in RFC7714) Key boosting with 2^16 SHA-512 iterations, whitening keys are zeroised HKDF: SHA512 (RFC5869) with key including key.
Key boosting	Used by **SF.MESSAGES** and **SF.CALLS** Key boosting involves executing a number of consecutive hash operations; Key boosting utilizes 2^16 SHA-512 iterations.
SIPS	Used by **SF.CHANNEL** for the connection to the SIP server Employs TLS SIPS is SIP [RFC3261] that runs over Transport Layer Security (TLS) [RFC5246]. Note that the security relies on TLS connection and not on SIP or MD5.
TLS	Used by **SF.CHANNEL** TLS 1.2 Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 Elliptic curve: secp384r1 • Used for the DPS webAPI connection • Used for the DCS webAPI connection • Used for the SIP Server DCS connection
AES CFB	Used by **SF.PUSH** AES-256 in CFB Mode
X509 Certificates	Used by **SF.CHANNEL, SF.CALLS and SF.MESSAGES** for the TLS authentication • RSA 3072bits • SHA512
RNG	Random number generation is using the CTR_DRBG algorithm on iOS (TOE environment).

The TOE relies on two open source projects for encryption technology:

- **mbedTLS**. This is an Open Source cryptography library that provides a wide variety of algorithms. The TOE uses mbedTLS's AES-256, SHA1, SHA2, MD5 and RSA, as well as its TLS implementation. Dencrypt has extended mbedTLS's functionality by Dynamic Encryption. See also https://tls.mbed.org/
- **libSRTP**. The libSRTP library is an open-source implementation of the Secure Real-time Transport Protocol (SRTP) originally authored by Cisco Systems, Inc. It implements AES-256 and SHA1. Dencrypt has extended libSRTP's functionality by Dynamic Encryption. See also http://srtp.sourceforge.net/srtp.html
- **decaf**. Elliptic curve operations are provided by the decaf library: X448, EdDSA448 and conversion from EdDSA key to ECDH key format.

8 Abbreviations and references

8.1 Abbreviations

AES	Advanced Encryption Standard
AES-CM	AES – counter mode
CBC	Cipher Block Chaining
CC	Common Criteria
CM	Counter Mode
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CUG	Closed User Group
DCM	Dencrypt Certificate Manager
DCC	Dencrypt Control Center
DCS	Dencrypt Communication Server
DDB	Dencrypt DataBase
DES	Data Encryption Standard
DH	Diffie-Hellman key exchange
DPS	Dencrypt Provisioning Server
DTLS	Datagram Transport Layer Security
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
GCM	Galois/Counter Mode
HMAC	Keyed-Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
MDM	Mobile Device Management
OSP	Organisational Security Policy
PKI	Public Key Infrastructure
PP	Protection Profile
RSA	Acronym for Rivest, Shamir, Adleman, the creators of the RSA algorithm
RTP	Real-Time Transport Protocol
SAR	Security Assurance Requirement
SAS	Short Authentication String
SDP	Session Description Protocol
SFP	Security Function Policy
SFR	Security Functional Requirement
SIM	Subscriber Identification Module

SIP	Session Initiation Protocol
SMS	Short Message Service
SMS-C	Short Message Service Center
SRTP	Secure Real-time Transport Protocol
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
VoIP	Voice over IP
X3DH	Extended Triple Diffie-Hellman
ZRTP	Zimmermann Real-time Transport Protocol

8.2 References

[CC] Common Criteria for Information Technology Security Evaluation. Part 1: Introduction and general model, April 2017, Version 3.1 Revision 5, CCMB-2017-04-001; Part 2: Security functional components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-002; Part 3: Security assurance components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-003.

[CEM] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, April 2017, Version 3.1 Revision 5, CCMB-2017-04-004.

[cPPND] Collaborative Protection Profile for Network Devices, Version 2.1, 24-Sept-2018.

[DR] The Double Ratchet Algorithm, Revision 1, 2016-11-20, https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf

[FIPS180-4] Federal Information Processing Standards Publication 180-4, Secure Hash Standard (SHS), 2012 March.

[FIPS186-4] Federal Information Processing Standards Publication 186-4, Digital Signature Standard (DSS), July 2013.

[FIPS197] Federal Information Processing Standards Publication 197, Advanced Encryption Standard (AES), November 26, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[ISO10118] ISO/IEC 10118-3:2018, October 2018, Information technology – Security Techniques – Hash-functions – Part 3: Dedicated hash-functions

[LIME] Linphone Instant Message Encryption v2.0, Version 1.0, March 6, 2019. https://gitlab.linphone.org/BC/public/lime/blob/master/lime.pdf

[NIST SP 800-38A] NIST Special Publication 800-38A 2001 Edition, NIST Special Publication 800-38A 2001 Edition, Recommendation for Block Cipher Modes of Operation. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

[NIST SP 800-38D] NIST Special Publication 800-38D, November 2007, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

[NIST SP 800-90A] NIST Special Publication 800-90A, June 2015, Recommendation for Random Number Generation Using Deterministic Random Bit Generators.
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

[PKCS1v2.1] PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories June 14, 2002
https://www.teletrust.de/fileadmin/files/oid/oid_pkcs-1v2-1.pdf

[RFC2104] HMAC: Keyed-Hashing for Message Authentication, February 1997.

[RFC2818] HTTP over TLS, May 2000

[RFC3261] SIP: Session Initiation Protocol, June 2002

[RFC3711] The Secure Real-time Transport Protocol (SRTP), March 2004

[RFC3830] MIKEY: Multimedia Internet KEYing, Ericsson Research, August 2004

[RFC5246] The Transport Layer Security (TLS) Protocol, Version 1, August 2008

[RFC5289] TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM), August 2008.

[RFC5763] Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS), May 2010

[RFC5764] Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)

[RFC5869] HMAC-based Extract-and-Expand Key Derivation Function (HKDF), May 2010

[RFC6125] Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS), March 2011.

[RFC6347] Datagram Transport Layer Security Version 1.2, January 2012

[RFC7714] AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP), December 2015

[RFC7748] Elliptic Curves for Security, January 2016

[RFC8032] Edwards-Curve Digital Signature Algorithm (EdDSA), January 2017

[X3DH] The X3DH Key Agreement Protocol, Revision 1, 2016-11-04
https://signal.org/docs/specifications/x3dh/x3dh.pdf

[Dynamic] Patent application WO 2013/060876 A1.
https://patents.google.com/patent/WO2013060876A1