Encryption Module Usage Guidelines

Certifiable Encryption

Document no.: GEC-US-0081
Author: Galleon Embedded Computing
Revision: 1.0.9
Date: May 15, 2026

Preface

Conventions

IEC prefixes are used to separate between decimal and binary multiples.

- Decimal prefixes k (kilo), M (mega), G (giga), and T (tera) denote 10^3, 10^6, 10^9, and 10^{12} respectively.
- Binary prefixes Ki (kibi), Mi (mebi), Gi (gibi), and Ti (tebi) denote 2^{10}, 2^{20}, 2^{30}, and 2^{40} respectively.

Dates printed in numerical format are in the order day month year. E.g. 24.01.10 (or 24.01.2010) denotes January 24th, 2010.

Copyright

Limited Liability

Galleon Embedded Computing assumes no liability arising from any use of information provided within this document.

Revision History

Rev.	*Date*	*Changes*
1.0.0	07-Jan-21	First draft
1.0.1	13-Apr-22	Added section on System Shutdown
1.0.2	27-Apr-22	Added section on Protocols
1.0.3	28-Apr-22	Updated section on Protocols
1.0.4	12-May-22	Added document number
1.0.5	21-Jun-22	Improved clarity of evaluated configuration
1.0.6	14-Jul-22	Minor corrections to section on Shutdown
1.0.7	19-Jun-23	Updated sections on SNMP Interface, TLS Certificates
1.0.8	21-Oct-25	Added Disable SNMP menu item
1.0.9	15-May-26	Clarify password constraints and purpose

Note: Revision numbers are on the form X.Y.Z, where X indicates the major revision number, Y indicates minor revision number, and Z indicates revision number for editorial changes that do not alter the technical content of this document.

Document Definitions

List of keyword definitions for document automation. Variables set at time of document creation.

Keyword:

Subject	Certifiable Encryption
Title:	Encryption Module Usage Guidelines
Product:	Encryption Module
Version:	1.0.9
Author:	aknowles@galleonec.com
Company:	Galleon Embedded Computing

Contents

1 Glossary

EM	(Hardware) Encryption Module
DAR	Data at Rest
RDM	Removable Data Module
DEK	Data Encryption Key
EDEK	Encrypted Data Encryption Key
KDB	Key Database
TLS	Transport Layer Security
SSH	Secure Shell
SNMP	Simple Network Management Protocol
CA	Certificate Authority
KEK	Key Encryption Key

2 Overview

This document describes how to safely provision and use an Encryption Module (EM) for hardware encryption of Data at Rest. The EM is a component of a larger system (XSRS/G1S) providing storage of and access to Data at Rest, but for security there is strictly limited interaction between the EM and the rest of the system, and the user is able to interact with the EM directly.

The system models evaluated with EM support are:

XSR with Intel Xeon E3-1505Lv6 (Kaby Lake)

XSR with Intel Xeon E-2276ME (Coffee Lake)

XSR with Intel Xeon W-11865MRE (Tiger Lake)

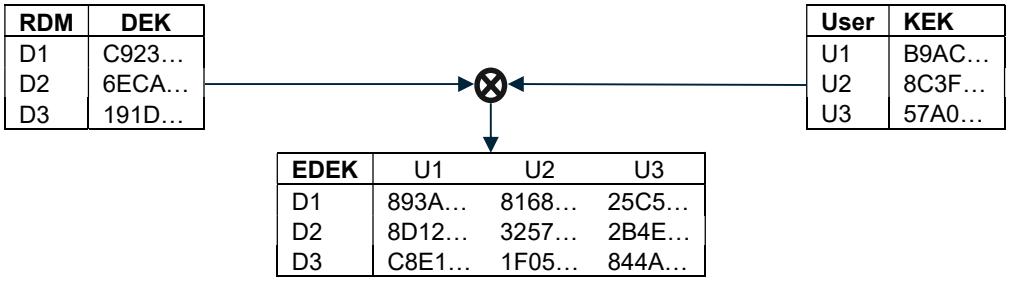
G1 with Intel Atom C2758 (Rangeley)

2.1 Conceptual Overview

The EM has two modes of operation: the provisioning mode and the deployed mode. While in provisioning mode the EM can be configured and prepared for deployment by authorised users with sufficient permissions. Once deployed, the EM is a read-only device that only supports an authentication process and some control and status signalling.

Provisioning mode must only be used while the XSRS/G1S is in a secure environment.

The EM is designed to support multiple Removable Data Modules (RDMs), where each RDM that the EM will need to unlock may have a unique Data Encryption Key (DEK). These DEKs are encrypted and stored in a Key DataBase (KDB) on the EM. The KDB is uploaded by the admin or crypto users while the EM is in a secure environment, and cannot be modified while the EM is deployed. The KDB contains a set of users and a set of RDMs, and a unique, encrypted DEK (EDEK) for each user that has access to each RDM.



In order to unlock the RDM and allow access to the data stored on it the operator must provide a username and passphrase to the EM. This will be referred to as the authentication process. The EM will authenticate the user within the KDB and, if there is an EDEK for that user for the current RDM, to decrypt the DEK and unlock the RDM.

There are three methods for providing authentication information to the EM, each of which can be enabled or disabled by the admin user during provisioning:

1. TLS connection (encrypted and mutually authenticated) over dedicated Ethernet.

2. SSH connection (encrypted and mutually authenticated) over dedicated Ethernet.

3. Serial terminal over RS-232.

The TOE's evaluated configuration requires that the administrator configure the TOE during provisioning to enable local authentication (serial terminal over RS-232) in the Authentication Methods sub menu (see 3.1.4). Section 2.2 below further touches upon this evaluated configuration.

There are three fixed Linux user accounts for the EM that are used to log in to the EM over either SSH or serial, if settings allow. These users are:

1. admin: for configuring the EM while in a secure environment.

2. crypto: for updating the KDB on the EM while in a secure environment.

3. operator: for attempting authentication with the KDB.

Note that if the TLS authentication method is used, or the SSH method with appropriate SSH keys or certificates, then the operator Linux user is not prompted to log in, and the operator can proceed directly to the KDB authentication process. The constraints on passwords are described in Section 3.1.3: Passwords Sub Menu.

For authentication to unlock the RDM there is a distinct set of users in the KDB and this is not a fixed set. These users are distinct from the Linux user accounts and have independent usernames and passphrases. The KDB users are only used for authentication when attempting to unlock an RDM and cannot be used to log in to Linux on the EM or perform configuration changes.

The KDB itself is created outside of the EM and uploaded to the EM by the admin or crypto Linux user account. The creator of the KDB decides which users are present in the KDB and which RDMs each of those users will be able to unlock. The KDB stores individual RDM DEKs encrypted as EDEKs. Multiple copies of an RDM's DEK are encrypted, one per authorized KDB user of that RDM. The entire KDB is also encrypted and signed by the creator, and the EM must be configured with the appropriate KDB certificate to be able to decrypt and validate the KDB before authorization is attempted.

2.2 Protocols

In addition to the RS-232 console interface, the EM supports the following network protocols: SSH, SCP, SFTP, TLS, NTP, DHCP and SNMP. In the default configuration only TLS and SNMP are enabled when not in provisioning mode, and additionally SSH is enabled while in provisioning mode.

Administrators can choose to enable or disable authentication via TLS and SSH in the Authentication Methods sub menu (see 3.1.4). This will affect both provisioning and deployed modes.

SSH is always enabled when in provisioning mode. When SSH is used as an Authentication Method in deployed mode it cannot be used for remote administration, only for authentication.

File transfer protocols SCP and SFTP are only enabled temporarily while in provisioning mode when triggered by an administrator's explicit action, to allow transferring files to the EM such as certificates, public keys, a KDB, or a software update.

DHCP is only enabled when the IP address is set to "dhcp" in the Network Configuration sub menu (see 0) and will be disabled when a static IP address is configured.

NTP is only enabled when a valid NTP server is specified in the Date and Time sub menu (see 3.1.2) and will be disabled when the NTP server is set to "none".

SNMP is enabled by default. It can be disabled in the Network Configuration sub menu (see 0).

The IP address of the user connecting to the EM can be constrained to limit it to the local subnet, or even to a single IP address, in the Network Configuration sub menu (see 0). This constraint can be separately specified for administrative tasks over SSH and for authentication attempts over SSH or TLS.

Provisioning mode must only be enabled when both the unit and the administrator are within the same local secure environment, and, if network access is used during provisioning, the network should either be directly connected between the unit and the administrator's PC, or connected to a secure, private LAN.

All provisioning activities can be performed locally over the RS-232 port with two exceptions:

1. KDBs larger than 12 KiB must be transferred over the network port (see 3.1.6).

2. EM software update packages must be transferred over the network port (see 3.1.7).

In the evaluated configuration RS-232 is enabled for authentication and authentication is performed locally over the serial connection.

The FDEEE and FDEAA Protection Profiles do not consider nor include networking protocols as part of the security functional requirements, and as a result, do not include any requirements for addressing these protocols.

As such, as per the protection profiles, these protocols have not been examined as part of the required assurance activities and consequently the evaluation can make no claims about the networking protocols on the EM.

In other words, network security is out of scope for the evaluation.

The customer must consider the impact of using both remote administration and authentication methods over the network in light of their planned deployment scenario, and factor this into their risk management decision. Galleon strongly recommends that any network access be limited to a segregated private network in both provisioning and deployed modes.

3 Provisioning

A factory configured EM must be provisioned by the admin user before being used. The EM has a discrete input that must be set to allow provisioning to occur. The admin user may connect to the EM using the RS-232 port or by over SSH to the default IP address 192.168.1.110 (on subnet /24). The factory configured password for the admin user is: admin

In the evaluated configuration, provisioning occurs over the RS-232 port.

The first step in provisioning the EM is changing the factory configured passwords.

Upon logging in, the admin user will be shown a configuration menu.

3.1 Main Menu

```
                    Galleon Embedded Computing
                 Encryption Module Configuration

 000 Main Menu

1 MN Network Configuration                          192.168.1.110
2 MD Date and Time
3 MP Passwords
4 MM Authentication Methods
5 MC Certificates
6 DB Update KDB                                     f0d43b7ed7501ced8a9f4ff
7 MU Update or Sanitize                             4.1.0
8 RE Sanitize and Restore Factory Defaults
9 MR Diagnostics

U Go up                  T Go to top                J Jump to
S Save                   R Reboot                   X Exit
```

The crypto user will be shown a reduced version of the configuration menu:

```
                    Galleon Embedded Computing
                 Encryption Module Configuration

 000 Main Menu

6 DB Update KDB                                     f0d43b7ed7501ced8a9f4ff
8 RE Sanitize and Restore Factory Defaults
9 MR Diagnostics

U Go up                  T Go to top                J Jump to
S Save                   R Reboot                   X Exit
```

3.1.1 Network Configuration sub menu

The network configuration menu allows the admin user to change the EM network settings. It is strongly recommended that the network environment allows for the EM to have the same network settings during both provisioning and deployment.

IP Address	Static IP address (IPv4) of the EM, can be set to DHCP
Netmask	Subnet mask, typically 255.255.255.0
Default Gateway	Gateway for routing out of subnet, can be none
Nameserver	Nameserver for DNS, can be none
Admin IP Filter (SSH)	IP address range (in CIDR format) for allowed admin user SSH client, can be none (to allow any IP)
Crypto IP Filter (SSH)	IP address range (in CIDR format) for allowed crypto user SSH client, can be none (to allow any IP)
Operator IP Filter (TLS & SSH)	IP address range (in CIDR format) for allowed operator user TLS/SSH client, can be none (to allow any IP)
Disable SNMP	If set to Yes, the EM will not respond to SNMP queries and the SNMP port will be blocked.

If the IP address of the EM is set to DHCP, the EM will use a DHCP client at boot to request a network address. This can make it difficult to determine the assigned IP address and then connect to the EM, but if the serial terminal is also available the assigned IP address can be discovered from the Show System Status command on the Diagnostics Menu.

The three filters can be optionally assigned to limit which source IP addresses can be used when connecting to the EM over SSH and TLS. They expect an IP in CIDR format, such as 192.168.1.0/24, which would limit source IPs to the default subnet.

Note that there are other limitations on when SSH and TLS connections for the three Linux users are permitted.

3.1.2 Date and Time sub menu

NTP Server	IP address of NTP server for setting current time at boot, can be none
Default Date	Default date and time to set at every boot

For robustness, the EM does not contain a battery and so is not able to remember the current time when losing power. The system can be configured with a default date and time to be used upon boot, and can optionally be configured with the IP address of an NTP server to retrieve the current time from when booting.

The current time is only important to the EM when validating TLS certificates. If new TLS certificates are generated and upload to the EM, it will be necessary to update the default time so that it is within the certificate validity period.

3.1.3 Passwords sub menu

Admin Password	Linux admin user password

Crypto Password	Linux crypto user password
Operator Password	Linux operator user password
SNMPv3 Password	Password for authenticate and encrypted SNMP v3 protocol

The admin user should change all three Linux user passwords when configuring the EM. Passwords are salted, hashed and stored in the standard Linux shadow format. Linux user passwords may be up to 256 characters in length and may consist of uppercase/lowercase letters, numbers, the space character, and the following special characters: !"#$%&/_\()~'=?*+-<>;[]:{}.|,@^

These passwords pertain to logging in to the EM as a Linux user or to access the SNMP service. None of these passwords are used during authentication. The difference between Linux users and KDB users is described in section 2.1 Conceptual Overview.

The SNMPv3 password is used for authenticated and encrypted SNMP communication with the EM. See the SNMP chapter.

3.1.4 Authentication Methods sub menu

Allow auth over TLS	Allow mutually authenticated TLS connections to perform KDB authentication
Allow auth over SSH	Allow Linux operator user to connect via SSH to perform KDB authentication
Allow auth over Serial	Allow Linux operator user to login via RS-232 to perform KDB authentication
Allow legacy key loading over TLS	Allow mutually authenticated TLS connections to directly transfer DEKs like earlier EM versions

The admin user can select any combination of the four authentication methods.

If neither of the TLS options are enabled, the TLS server will not be started, and the TLS port (TCP 9000) will be blocked.

If the SSH option is not selected, the Linux user operator will not be allowed to log in via SSH. If the EM is not in a secure area, the SSH server will not be started and the SSH port (TCP 22) will be blocked.

If the serial option is not selected, the Linux user operator will not be able to authenticate when logged in via serial. If the SSH option is also not selected, the Linux user operator will not be able to log in to the EM at all.

If the legacy option is selected, it is possible for the TLS client to directly transfer the DEK to decrypt the RDM. This method bypasses the KDB and KDB user authentication. The DEK is encrypted by the TLS connection, but is not otherwise protected by a password.

In the evaluated configuration, authentication is performed over the RS-232 serial port.

3.1.5 Certificates sub menu

The certificates sub-menu contains further sub-menus for TLS and SSH certificates, and allows updating the KDB certificates:

```
                        Galleon Embedded Computing
                      Encryption Module Configuration

 500 Certificates

1 MT TLS Certificates
2 MK SSH Public Keys
3 MS SSH CA Certificates
4 KA KDB Certificate A                          34:04:68:C5:5B:B5:90:EA
5 KB KDB Certificate B

U Go up                  T Go to top            J Jump to
S Save                   R Reboot               X Exit
```

KDB Certificate A	Upload an x509 certificate for validating the KDB signature
KDB Certificate B	Upload an alternate x509 certificate for validating the KDB signature

The KDB itself must also be signed by a private key, to be validated by an x509 certificate that is distributed to all of the EMs that will need to read a particular KDB. Two validation certificates can be uploaded, to allow for graceful rollover from one to the other.

The KDB certificate can only be updated by the admin user, while the KDB itself can be updated by both the admin user and the crypto user. This allows for a model where the EMs are initially provisioned in a highly secure area, by the admin user, and then the KDB can later be updated in other secure areas by a trusted subordinate crypto user. Only the user with access to the KDB private signing key (i.e. the admin user) can generate a new KDB that will be accepted by the EM, but once created, a new KDB can be distributed to be installed on EMs.

This also allows for a KDB to be created that can only be used by a specific EM – if each EM is given a unique KDB certificate, then the creator of the KDB can choose which KDB private key to sign with, and only the corresponding EM will be able to use the KDB.

TLS Certificates sub menu

Generate new EM keypair	Generate a new public and private key internally for TLS connections
Replace EM keypair	Upload an externally generated private key for TLS connections
Generate EM Certificate Request	Generate a x509 certificate request for this EM using the current (possibly unsaved) private TLS key
EM Certificate	Upload an x509 certificate for the EM

Root CA Certificate A	Upload an x509 certificate for the root authority that validates TLS clients
Root CA Certificate B	Upload an alternate x509 certificate for the root authority that validates TLS clients
Certificate Revocation List	Upload a certificate revocation list to invalidate TLS clients

The TLS connection is mutually authenticated, which means that both the EM (the TLS server) and the operator (the TLS client) present x509 certificates to each other, and both must be validated. The EM configuration supports storing up to two root CA certificates, and the TLS client certificate must validate against one of them.

The factory configuration contains a default self-signed certificate for both the EM and the example TLS client, but these should be replaced when configuring the unit.

The EM can either generate a unique public and private key itself, in which case the private key is never exposed, or the EM can be configured with an externally generated keypair. The latter can be useful if configuring a fleet of identical EMs, for example, but care must be taken to keep the private key secret.

If using an externally generated keypair for the EM, the TLS certificate may be self-signed. In this case, the TLS clients are configured with a copy of the EM certificate to trust directly, rather than a root CA certificate. Similarly (but independently), the EM can trust either two root CA certificates that can validate client TLS certificates, or a self-signed client certificate can be directly trusted by uploading it as a root CA certificate.

However, to properly manage a set of multiple EMs and a set of multiple TLS clients, it is best to generate a root CA certificate that can then sign both EM certificate requests and TLS client certificate requests. It can also be a good idea to generate a second root CA certificate and install it to the EMs but not use it for signing yet. This allows for a graceful root certificate rotation period, where the first root CA certificate is still trusted while not being used for signing, and EMs and TLS clients are gradually issued new certificates signed by the second root CA while still being able to interoperate.

See the TLS Certificate section for guidance on how to set up a root CA and issue certificates to EMs and TLS clients.

SSH Public Keys sub menu

Generate new EM keypair	Generate a new public and private key internally for SSH connections
Replace EM private key	Upload an externally generated private key for SSH connections
Show EM Public Key	Display the EM SSH public key, for adding to SSH client known hosts files
Admin Public Key A	Upload an SSH public key for the linux admin user
Admin Public Key B	Upload an alternate SSH public key for the linux admin user
Crypto Public Key A	Upload an SSH public key for the linux crypto user
Crypto Public Key B	Upload an alternate SSH public key for the linux crypto user
Operator Public Key A	Upload an SSH public key for the linux operator user
Operator Public Key B	Upload an alternate SSH public key for the linux operator user

SSH connections are also mutually authenticated, but unlike TLS the more common configuration with SSH connections is to have either pre-shared public keys between peers, or to trust a peer's public key on first connection and then remember it for future connections (trust-on-first-use). This model can be difficult to manage when there are many peers, or when multiple peers share a common identify, like an IP address. The next section describes an alternative using SSH Certificates that could be more appropriate but is only supported by some SSH clients.

The factory configuration contains a unique keypair generated for the EM at the factory. This can be replaced by a freshly generated keypair, or an externally generated keypair can be uploaded. The latter can be useful if configuring a fleet of identical EMs, for example, but care must be taken to keep the private key secret.

The EM public key can be displayed, so that SSH clients can be configured to trust the key, and the trust-on-first-use model can be disabled in clients if desired.

Each of the three Linux users support uploading two trusted SSH client public keys. If the EM is going to need to interoperate with more than two operator SSH clients, then either the clients must be configured with identical keypairs, or SSH Certificates should be used instead.

SSH CA Certificates sub menu

SSH CA Public Key A	Upload an SSH certificate for the root authority that validates SSH clients
SSH CA Public Key B	Upload an alternate SSH certificate for the root authority that validates SSH clients
EM SSH Certificate	Upload an SSH certificate for the EM
Key Revocation List	Upload a public key revocation list to invalidate SSH clients

SSH Certificates can be used to automatically trust SSH clients who can present an SSH certificate that can be validated with one of the two SSH CA public keys that the EM has been configured with. This allows any number of SSH clients to have unique keypairs and still be strongly authenticated upon connection.

Similarly, the SSH CA can sign the EM public key (shown from the SSH Public Keys sub menu) and generate an SSH certificate for the EM, which can be uploaded to the EM. This will allow SSH clients that trust the SSH CA public key to trust the EM public key without having seen it before.

SSH client certificates that are no longer trustworthy (even though they are signed by a still-trusted CA) can be revoked by adding them to a key revocation list and uploading the KRL to the EM.

3.1.6 Update KDB

The KDB itself can be updated by either the admin or crypto Linux users. When the Update KDB menu item is selected, the user can either transfer the KDB binary file to the EM over the Ethernet port using the scp or sftp protocols, or the user can Base64 encode the binary file and paste the resulting text directly into the configuration menu. The Base64 option is only permitted for KDB files less than 12 KiB and is the only way to upload a KDB over the RS-232 serial connection.

In the evaluated configuration, the KDB is updated via the RS-232 serial connection.

Upon upload, the EM will validate the KDB against either of the two current KDB certificates (if the certificates are freshly updated and have not yet been saved to the EM, they are still used for validation in this instance. This allows the admin to update the certificates and KDB in a single configuration cycle).

If the KDB does not validate correctly it will not be stored on the EM, and the previous KDB will not be replaced.

3.1.7 Update or Sanitize sub menu

Update EM Software (keep configuration)	Upload an EM update package, which will upgrade to a new version while keeping the current configuration
Sanitize and Update EM Software	Upload an EM update package, which will erase the current configuration and upgrade to a new version
Sanitize and Restore Factory Defaults	Erase the current configuration and restore factory default configuration

Galleon will notify customers directly as software update packages applicable to the Encryption Module become available.

Updates can only be applied by the admin user. When updating, the user can decide whether to keep the current EM configuration with the new EM software version or to securely erase the current configuration and start the new EM software version with factory default configuration.

The update will be automatically validated by the Encryption Module and will not continue if the embedded digital signature in the update package cannot be verified.

3.1.8 Sanitize and Restore Factory Defaults

The admin and crypto users can securely erase the current configuration to revert to the factory default configuration with the current EM version.

Erasing the configuration will securely erase the entire configuration partition, which is used to store network configuration, passwords, private keys, public keys, certificates, and the KDB itself. It will be necessary to fully reconfigure the EM after erasing the current configuration.

3.1.9 Diagnostics sub menu

Report Configuration	Display current configured options or file hashes where appropriate
Show System Log	Display the system message log
Show System Status	Display current system status and diagnostic information

The diagnostics sub menu can display current system status, a configuration summary, and a system log, to help aid in troubleshooting. The system log is volatile only and will be lost upon reboot.

4 TLS Certificates

If the operator is permitted to attempt authentication over TLS then TLS certificates must be issued and configured by the administrator, both to each EM in operation and to each TLS client terminal that will communicate with an EM on behalf of the operator.

In the evaluated configuration, authentication over TLS is not used.

There are two methods of configuring TLS certificates:

1. Root CA Signed Certificates, where every EM and TLS client are given unique certificates signed by a (locally administered) central authority and can all interoperate with each other.
2. Self-Signed Certificates, where a single EM and TLS client are given each other's certificates and can only communicate between themselves.

The EM client tools package includes scripts to automatically create, sign and upload certificates to the EM, in both certificate configurations.

4.1 Root CA Signed Certificates

Using root-signed certificates means there is a single trusted root certificate that is installed on all communicating Encryption Modules and external key providers, and that all of these have their individual certificates signed by the trusted root. This is a more flexible configuration.

If you do not already have a root certificate, you will need to create one.

First, create a certificate configuration file containing the information to be placed into your root certificate:

```
RANDFILE               = $ENV::HOME/.rnd

[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no

[ req_distinguished_name ]
C                      = NO
ST                     = Oslo
L                      = Oslo
O                      = Galleon Embedded Computing AS
OU                     = RD
CN                     = TLS Authentication CA
emailAddress           = support@galleonec.com
```

Save this as ca-cert.conf.

Second, create a self-signed root certificate like so:

```
openssl ecparam -genkey -name secp384r1 -out ca-key-a.pem
openssl req -new -config ca-cert.conf -key ca-key-a.pem -out ca-req-a.pem
openssl req -x509 -in ca-req-a.pem -days 36500 -key ca-key-a.pem > ca-cert-a.pem
```

Third, optionally, create a second root certificate to allow for a graceful transition away from the first one:

```
openssl ecparam -genkey -name secp384r1 -out ca-key-b.pem
```

```
openssl req -new -config ca-cert.conf -key ca-key-b.pem -out ca-req-b.pem
openssl req -x509 -in ca-req-b.pem -days 36500 -key ca-key-b.pem > ca-cert-b.pem
```

Keep ca-cert-b.pem available to be uploaded to EMs under configuration, but move ca-key-b.pem to a safe location, preferably in cold storage.

Next, create a CA configuration file:

```
RANDFILE        = $ENV::HOME/.rnd

[ ca ]
default_ca      = CA_default            # The default ca section

[ CA_default ]
dir             = .                     # top dir
database        = $dir/index.txt        # index file.
new_certs_dir   = $dir                  # new certs dir

certificate     = $dir/ca-cert-a.pem    # The CA cert
serial          = $dir/serial           # serial no file
private_key     = $dir/ca-key-a.pem     # CA private key

default_days    = 36500                 # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = sha384                # md to use

policy          = policy_any            # default policy
email_in_dn     = no                    # Don't add the email into cert DN

name_opt        = ca_default            # Subject name display option
cert_opt        = ca_default            # Certificate display option
copy_extensions = none                  # Don't copy extensions from request
unique_subject  = no                    # Allow new certificates with same CN

[ policy_any ]
countryName             = supplied
stateOrProvinceName     = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Save the file as ca.conf.

Create an empty certificate database file called index.txt and a next serial number file called serial using the following commands:

```
touch index.txt
echo 01 > serial
```

For each EM to be configured, choose to generate a new TLS key pair (type JCG from any menu) and then generate an EM certificate request (type JCR). This will print a certificate request to the terminal. Copy and paste this request into a text file and save it as em-req.pem in the directory where the CA was created.

Now sign the EM certificate using this command:

```
openssl ca -config ca.conf -in em-req.pem -out server-cert.pem
```

Upload the EM certificate to the EM by selecting the EM Certificate menu item (type JCE) and either pasting the contents of server-cert.pem into the terminal, or copying the file itself using a command like:

```
scp server-cert.pem admin@192.168.1.110:/tmp
```

Upload the CA certificate to the EM by selecting the Root CA Certificate A menu item (type JCA) and either pasting the contents of ca-cert-a.pem into the terminal, or copying the file itself using a command like:

```
scp ca-cert-a.pem admin@192.168.1.110:/tmp
```

Save the changes to the EM configuration and reboot the EM for the changes to take effect (type SR).

For each Authentication Client the private key and certificate request can either be generated on the Authentication Client itself or centrally, by the CA.

First, create a certificate configuration file containing the information to be placed into the certificate:

```
RANDFILE               = $ENV::HOME/.rnd

[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no

[ req_distinguished_name ]
C                      = NO
ST                     = Oslo
L                      = Oslo
O                      = Galleon Embedded Computing AS
OU                     = RD
CN                     = TLS Authentication Client #1
emailAddress           = support@galleonec.com
```

Save this as client.conf. Each client certificate issued by the same CA will need a unique common name, so the CA = TLS Authentication Client #1 will need to be updated.

Second, generate a private key and a certificate request for the client:

```
openssl ecparam -genkey -name secp384r1 -out client-key.pem
openssl req -new -config client.conf -key client-key.pem -out client-req.pem
```

Transfer client-req.pem to the CA if necessary.

Third, generate the certificate for the client:

```
openssl ca -config ca.conf -in client-req.pem -out client-cert.pem
```

Transfer ca-cert-a.pem (renamed to server-cert.pem as the client tools expect), client-cert.pem, and client-key.pem (if necessary) to the Client Authentication device.

For example:

```
cp ca-cert-a.pem ../gec-crypto-key-client/linux/key-exchange/server-cert.pem
cp client-cert.pem ../gec-crypto-key-client/linux/key-exchange/
cp client-key.pem ../gec-crypto-key-client/linux/key-exchange/
```

4.2 Self-Signed Certificates

If the deployment is such that only a single Client Authentication device and a single EM will communicate with each other, it may be simpler to deploy self-signed certificates. In this case the private key for the EM must be generated outside the EM so that it can be used to self-sign the EM certificate.

First, generate both client and server private keys, using OpenSSL:

```
openssl ecparam -genkey -name secp384r1 -out client-key.pem
openssl ecparam -genkey -name secp384r1 -out server-key.pem
```

Second, create a certificate configuration file containing the information to be placed into your certificates:

```
RANDFILE               = $ENV::HOME/.rnd

[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no

[ req_distinguished_name ]
C                      = NO
ST                     = Oslo
L                      = Oslo
O                      = Galleon Embedded Computing AS
OU                     = RD
CN                     = Encryption Module
emailAddress           = support@galleonec.com
```

Save this as server.conf. Create a similar file and save it as client.conf.

Third, generate certificate requests for the client and server:

```
openssl req -new -config client.conf -key client-key.pem -out client-req.pem
openssl req -new -config server.conf -key server-key.pem -out server-req.pem
```

Fourth, sign the certificate requests in order to generate the client and server certificates:

```
openssl req -x509 -in client-req.pem -days 3650 -key client-key.pem > client-
cert.pem
openssl req -x509 -in server-req.pem -days 3650 -key server-key.pem > server-
cert.pem
```

(Where days is the number of days the certificate will remain valid.)

Upload the EM keypair to the EM by selecting the Replace EM keypair menu item (type JCK) and either pasting the contents of server-key.pem into the terminal, or copying the file itself using a command like:

```
scp server-key.pem admin@192.168.1.110:/tmp
```

Upload the EM certificate to the EM by selecting the EM Certificate menu item (type JCE) and either pasting the contents of server-cert.pem into the terminal, or copying the file itself using a command like:

```
scp server-cert.pem admin@192.168.1.110:/tmp
```

Upload the client certificate to the EM by selecting the Root CA Certificate A menu item (type JCA) and either pasting the contents of client-cert.pem into the terminal, or copying the file itself using a command like:

```
scp client-cert.pem admin@192.168.1.110:/tmp/ca-cert-a.pem
```

Save the changes to the EM configuration and reboot the EM for the changes to take effect (type SR).

Transfer server-cert.pem, client-cert.pem, and client-key.pem to the Client Authentication device.

For example:

```
cp server-cert.pem ../gec-crypto-key-client/linux/key-exchange/
cp client-cert.pem ../gec-crypto-key-client/linux/key-exchange/
```

```
cp client-key.pem ../gec-crypto-key-client/linux/key-exchange/
```

5 SSH

SSH communications should be secured by provisioning SSH public keys or certificates for the three Linux users of the EM. The EM uses 384-bit ECDSA type keys for SSH key exchange.

For the admin and crypto users, a public key or certificate functions as a second factor for authentication. The Linux user password is still required to log on, but the connecting client must also have the appropriate private key to gain access.

For the operator user, a public key or certificate replaces the Linux user password. This is because the only action available to the operator user is to perform the authorization process by providing a KDB username and password, and the Linux user password is superfluous.

In the evaluated configuration, authentication via SSH is not used.

Generate a public/private keypair for each of the admin, crypto and operator users. This should be done on the machines that will be used to connect to the EM over SSH, to keep the private keys isolated there.

```
ssh-keygen -o -b 384 -t ecdsa -C '' -N '' -q -f admin
ssh-keygen -o -b 384 -t ecdsa -C '' -N '' -q -f crypto
ssh-keygen -o -b 384 -t ecdsa -C '' -N '' -q -f operator
```

Each of these generates a private key file with no file extension (i.e. operator) and a public key file ending in .pub (i.e. operator.pub).

For each EM, generate a new SSH keypair on the EM using the Generate new EM keypair menu item (type JPG) and then show the EM SSH public key (type JPP). This will print the SSH public key to the terminal. Copy and paste this into a text file and save it as em-ssh-key-<serial>.pub, where <serial> is the serial number of the unit, for example.

Optionally, create an SSH CA public/private keypair in order to issue SSH certificates. This makes it possible to generate new SSH keypairs for EMs and authorization terminals and have them be automatically trusted by any device configured to trust the SSH CA public key.

```
ssh-keygen -o -b 384 -t ecdsa -C '' -N '' -q -f ssh-ca
```

The SSH CA can then issue SSH certificates when given an SSH public key:

```
ssh-keygen -s ssh-ca -I admin -n admin admin.pub
ssh-keygen -s ssh-ca -I crypto -n crypto crypto.pub
ssh-keygen -s ssh-ca -I operator -n operator operator.pub
ssh-keygen -h -s ssh-ca -I em-<serial> em-ssh-key-<serial>.pub
```

This generates certificate files ending in -cert.pub (i.e. operator-cert.pub). Note that the EM certificate must be generated with the host flag set (-h).

For each EM, upload the SSH CA public key (ssh-ca.pub) and the appropriate EM SSH certificate (em-ssh-key-<serial>-cert.pub) to the EM.

For each authorization terminal, add the SSH CA public key (ssh-ca.pub) to the active user's SSH known_hosts file, preceeded by "@cert-authority *":

```
printf "@cert-authority * " | cat - ssh-ca.pub >> ~/.ssh/known_hosts
```

Copy the operator private key, public key and certificate file to the authorization terminal, if they were not generated there. Then, when performing the authorization process, connect to the EM using SSH:

```
ssh -i ./operator -o StrictHostKeyChecking=yes operator@192.168.1.110
```

Where 192.168.1.110 is the configured IP address of the EM, and ./operator is the path to the operator private key file. The StrictHostKeyChecking option sets SSH to only trust keys already in the known_hosts file, eliminating the risk of the user choosing to trust a new host key when they should not.

The admin and crypto terminals should be configured in a similar fashion.

If not using SSH certificates, instead upload the admin.pub, crypto.pub and operator.pub files to each EM that will need to connect with over SSH. The EM supports only two public keys for each user – if this limits your deployment scenario then consider using SSH certificates, or securely distributing the operator private/public keypair to multiple authorization terminals.

6 Authorization

The authorization process consists of the following parts, in sequence:

1. The operator connects to the EM over TLS, SSH or serial. Depending on the configuration, this may involve logging in to the operator Linux user account using a password.

2. The operator is prompted to enter a username and password. For the SSH and serial methods, the EM does this prompting, while in the TLS case the prompt is performed by the authorization client program running on the authorization terminal.

3. The EM finds the user in the KDB users table and validates the password is correct by generating the user's KEK, encrypting a test value, and comparing the result to a stored test value in the KDB.

4. The EM uses the currently inserted RDM's serial number to find the appropriate EDEK for the user/RDM pair in the KDB. The EDEK is decrypted with the user's KEK and validated using the EDEK tag.

5. The EM starts the inline SATA crypto devices and transfers the DEK to them.

6. The EM removes all temporary values from memory – the DEK, KEK, password, etc.

In the evaluated configuration, authorization occurs over the RS-232 serial port.

If the authorization succeeds, the EM enters the keyed state and will not accept further authorization attempts. If authorization fails, the EM increments the authorization failure counter and allows for another authorization attempt. If the authorization failure counter reaches the limit of 5, the EM will automatically reboot.

At any point in time, the EM may receive a key erase command. This may come from a discrete input or over the network via SNMP. A key erase command will abort any in-progress authentication attempt, erase any DEKs from the inline SATA crypto devices, and return the EM to the unkeyed state. As this is asynchronous to the authorization process, it is possible for an operator to experience an authorization attempt fails even when the username and password were entered correctly.

If the RDM is not present at power-on or is removed, the EM will erase any DEK from the inline SATA crypto devices and enter the wait-disk state. The EM will not accept authorization attempts until an RDM is present.

If the EM does not pass BIT at power-on, or if BIT should fail at any time, the EM will erase any DEK from the inline SATA crypto devices and enter the BIT-fail state. The EM will not accept authorization attempts until the issue is resolved and BIT passes.

7 KDB

While resident on the EM the KDB cannot be updated or modified. It can only be replaced wholesale by the admin or crypto Linux users. The KDB must therefore be created on a different machine in a secure environment and signed using the KDB signing private key. The corresponding KDB signing certificate must be distributed to all of the EMs that are going to be configured with the KDB.

As the KDB is not created on the EM, creation of the KDB is out of scope for this document. Galleon provides example utilities with source code for KDB creation suitable to typical use cases.

The KDB is a compact binary file containing a table of users, a table of disks, and a mapping between the two tables. Each user-disk pair in the mapping contains an encrypted DEK (EDEK) for the disk, which can be decrypted only given the user's passphrase, which is not stored in the KDB. Once decrypted, the DEK can be used to decrypt the disk itself.

Users are identified by a username up to 16 bytes long, which must be unique, and the users table can contain up to 65535 users.

Disks are identified by a serial number, also up to 16 bytes long, which must be unique. The serial number is read automatically by the G1S from the RDM. Unique RDM serial numbers are programmed into each RDM sold by Galleon during production and cannot be changed.

The complete KDB file format is specified in the document "KDB Format Specification".

8 SNMP Interface

The Encryption Module includes a simple SNMP interface. The interface has three endpoints:

1. Key load status
2. Key erasure
3. Encryption Module reset

The SNMP interface is published in the `galleon` community with OID prefix `.1.3.6.1.4.1.8072.9999.9999`. The SNMP server supports SNMP protocol versions 1, 2c and 3.

SNMPv3 is configured with an encrypted, authenticated user ‘galleon’ with default password ‘galleonec’.

8.1 Key Load Status

The following command will fetch the key load status from the Encryption Module

```
snmpget -v 2c 192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.4
```

or

```
snmpget -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec"
192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.4
```

This command will return an integer. When interpreted as a 16-bit integer in little endian format (least significant byte first), a two byte status code is obtained. This status code is identical to that returned to the external key provider during a key transfer.

Byte 1	Byte 2	SNMP	Description
0x01	0x01	257	Ready: Waiting for authentication session
0x01	0x02	513	Waiting for RDM to be inserted
0x01	0x03	769	Waiting for BIT to complete
0x01	0x04	1025	Waiting for host system to finish booting
0x02	0x01	258	Error: Incorrect number of bytes transferred
0x02	0x02	514	Error: Client version is not supported by server
0x02	0x03	770	Error: Failed to transfer keys to crypto module
0x02	0x04	1026	Error: Crypto module already has keys - perform key wipe first
0x02	0x05	1282	Error: Crypto module reports unrecognised slave error
0x02	0x06	1538	Error: Crypto module reports unrecognised master error
0x02	0x07	1794	Error: Unable to transfer keys while key wipe is asserted
0x02	0x08	2050	Error: Unable to transfer keys while host module is reset
0x02	0x09	2306	Error: Unable to transfer keys to all RDM modules, check RDM

0x02	0x0A	2562	Error: Unable to communicate with crypto module
0x02	0x0B	2818	Error: Authentication attempt failed
0x02	0x0C	3074	Error: Unable to start authentication attempt, system not ready
0x02	0x0D	3330	Error: Unable to complete auth attempt, no longer ready
0x03	0x01	259	Success: Authentication attempt started
0x03	0x02	515	Success: Keys transferred to crypto module

Table 1: Key load status

8.2 DEK Erasure

The following command will trigger a DEK erasure.

```
snmpset -v 2c -c galleon 192.168.1.110 .1.3.6.1.4.1.8072.9999.9999.5 i 1
```

or

```
snmpset -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec" -c
galleon 192.168.1.110 .1.3.6.1.4.1.8072.9999.9999.5 i 1
```

This will wipe all DEKs from volatile memory on the EM and return it to the unkeyed state, ready for a new authorization attempt to occur.

Warning! Care should be taken not to perform a key erasure while the RDM is being actively written to. Erasing keys is similar to physically removing the RDM and may result in data corruption.

8.3 Encryption Module Reset

The following command will trigger a reset of the Encryption Module.

```
snmpset -v 2c -c galleon 192.168.1.110 .1.3.6.1.4.1.8072.9999.9999.6 i 1
```

or

```
snmpset -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec" -c
galleon 192.168.1.110 .1.3.6.1.4.1.8072.9999.9999.6 i 1
```

This will reset the dedicated crypto processors, wipe DEKs from volatile memory, and reboot the service processor on the EM.

Warning! Care should be taken not to perform a reset while the RDM is being actively written to. Resetting the Encryption Module is similar to physically removing the RDM and may result in data corruption.

8.4 System State Name

The following command will fetch the system state name from the Encryption Module.

```
snmpget -v 2c 192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.7
```

or

```
snmpget -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec"
192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.7
```

This command will return a string representing the Encryption Module system state. The possible state names and their descriptions are listed below in Table 2.

8.5 System State Description

The following command will fetch the system state description from the Encryption Module.

```
snmpget -v 2c 192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.8
```

or

```
snmpget -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec"
192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.8
```

This command will return a string describing the Encryption Module system state. The possible state names and their descriptions are listed below in Table 2.

State Name	State Description
START	Startup
WAIT_WIPE	Waiting for keywipe to deassert
BIT_RUN	Running BIT
BIT_FAIL	BIT failure
WAIT_RDM	Waiting for RDM
WAIT_DELAY	Waiting for host to boot
READY	Ready to key
AUTH_START	Key attempt starting
AUTH_READY	Key attempt in progress
AUTH_FAIL	Key failure
AUTH_OK	Key attempt succeeded
KEYED	Keyed
KEY_WIPE	Waiting for keywipe to deassert

Table 2: System states

8.6 RDM Serial Number

The following command will fetch the currently inserted RDM serial number.

```
snmpget -v 2c 192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.9
```

or

```
snmpget -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec"
192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.9
```

This command will return a string containing the RDM serial number. If there is no RDM present in the system, the string “(RDM not present)” will be returned. If there is an RDM inserted that has blank EEPROM such that the serial number cannot be read, the string “(RDM ID empty)” will be returned. Valid RDM serial numbers will not begin with “(“.

8.7 EM Serial Number

The following command will fetch the Encryption Module serial number.

```
snmpget -v 2c 192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.10
```

or

```
snmpget -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec"
192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.10
```

This command will return a string containing the EM serial number. An example serial number is “NAN203400943”.

8.8 EM Software Package Version

The following command will fetch the Encryption Module software package version number.

```
snmpget -v 2c 192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.11
```

or

```
snmpget -v 3 -u galleon -l authPriv -a SHA -A "galleonec" -x AES -X "galleonec"
192.168.1.110 -c galleon .1.3.6.1.4.1.8072.9999.9999.11
```

This command will return a string containing the EM software package version number. An example version number is “4.1.0”.

9 System Shutdown

The Encryption Module itself can be safely powered down without giving a shutdown command, as non-volatile memory is only updated when a save command is issued. The rest of the system, however, should be prepared for a shutdown before it is safe to remove the power supply.

The Encryption Module has no direct interface to the host system, by design. That means that in order to safely power down the system the operator must connect to the host system directly and issue a shutdown command. The host system interface is covered in more detail in the SW Encryption Layer Usage Guidelines, however the shutdown procedure is reproduced here.

1. If connecting directly to the XSRS/G1S though the locally attached keyboard and monitor, then

    a. Login as the root user (the default password is galleon).

    b. Run the following command in a terminal:
    poweroff

2. If connecting over SSH (to the default IP address of 192.168.100.101 on GbE port 0), run the poweroff command as part of the SSH command. This can be done as the user operator:
    ssh operator@192.168.100.101 poweroff

In the evaluated configuration, the host system is shutdown using the locally attached keyboard and monitor.

Once completed, the host system will typically transition to the compliant power-saving state G3 “mechanical off” within 10 seconds. Then the power supply can safely be removed from the system.

The power supply must be removed for the Encryption Module to enter the compliant power-saving state G3, mechanical off. The Encryption Module immediately transitions to the mechanical off state G3 when power is removed.

If the power supply is removed unexpectedly, the Encryption Module immediately transitions to the compliant power-saving state G3, mechanical off.

After transitioning out of the compliant power-saving state, when power is again provided to the Encryption Module, the authentication process must be completely restarted by providing a username and passphrase to the EM. Only once authentication succeeds will the RDM be unlocked, allowing access to the data stored on it.