galaxkey®
www.galaxkey.com

Galaxkey & nShield

HSM Integration

1.1.1

Galaxkey Limited
Fountain Court, 2 Victoria Square,
St Albans AL1 3TF
www.galaxkey.com

Table of Contents

Section 1. Version History

Version Number	Revision Date	Summary of Changes	Changed by
V 1.0	12 Oct 2020	Created initial document	OK
V1.1	4 Mar 2021	Updated to reflect latest HSM version	OK

Section 2. Copyright

Section 3. About Galaxkey

Galaxkey is a data protection company providing a portfolio of corporate data protection products to secure all data and support multinational data compliance regulations. Galaxkey is a global company with its headquarters in the UK. Our Galaxkey team has vast experience working with large multinational organisations across a variety of sectors and Galaxkey has select partners globally to provide local support and service to our customers across the globe. Our robust and experienced team offers exceptional customer service as well as quick response and turnaround times.

3.1. Disclaimer

The information contained in this document is confidential, privileged and only for the information of the intended recipient and may not be used, published or redistributed without the prior written consent of Galaxkey Ltd.

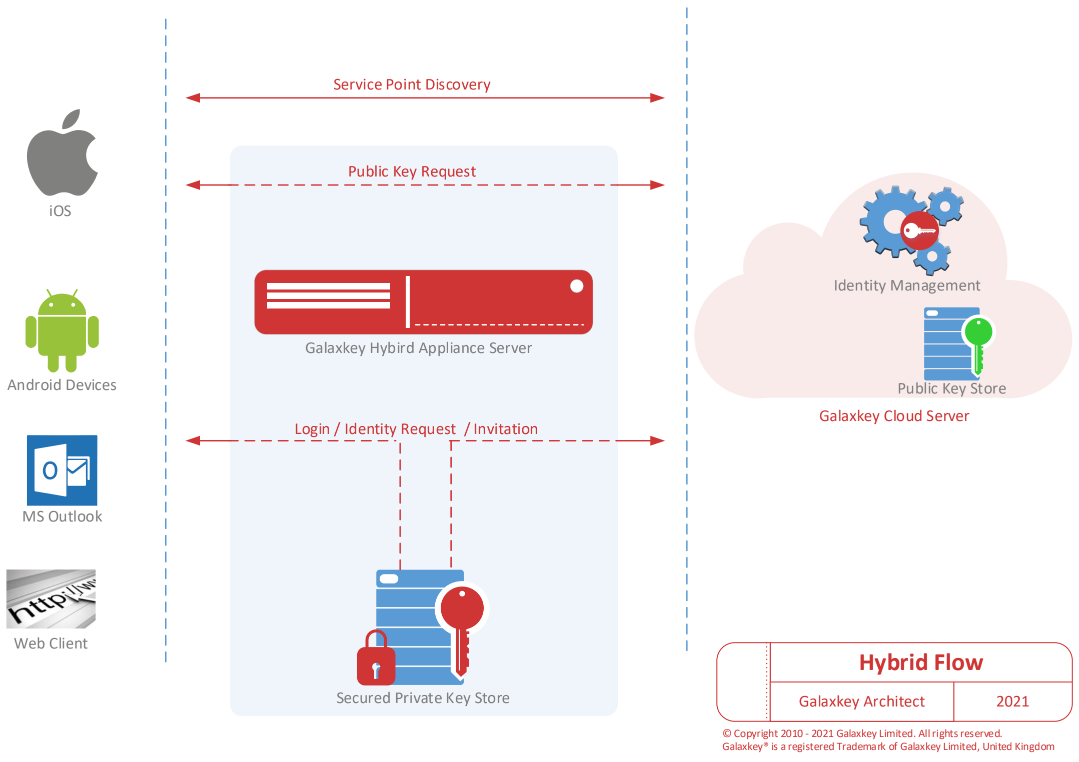
While every care has been taken in preparing this document, Galaxkey Ltd cannot be held responsible for any errors or omissions. The information in this document is subject to change without prior notice.

Galaxkey is a registered trademark of Galaxkey Ltd., registered in the U.K. and other countries. All other trademarks belong to their respective owners.

Section 4. Galaxkey Architecture

4.1.Architecture

As seen in the diagram below, there are 3 major components of Galaxkey architecture.


Service Point Discovery
Public Key Request
Login / Identity Request / Invitation
iOS
Android Devices
MS Outlook
Web Client
Galaxkey Hybird Appliance Server
Secured Private Key Store
Identity Management
Public Key Store
Galaxkey Cloud Server
Hybrid Flow
Galaxkey Architect
2021
© Copyright 2010 - 2021 Galaxkey Limited. All rights reserved.
Galaxkey® is a registered Trademark of Galaxkey Limited, United Kingdom

1. **Galaxkey Cloud server**: This is the centralised server for managing identities of individual users as well as managing corporate accounts.
2. **Galaxkey Hybrid appliance**: This is in-house server for an enterprise that opted for hybrid installation of Galaxkey. In this case the private keys of the domains managed by the appliance are stored on the appliance.
3. **Galaxkey Clients**: These are set of applications that use identities managed by server to facilitate the security of email and documents for a user. These include Outlook addin, Mobile device apps (iOS, Android, Windows Phone), Galaxkey for Windows that allows user to secure files / folders and share them securely, Mac client and Galaxkey Web Access.

4.2. Galaxkey Security

At the heart of the Galaxkey platform is user’s identity. The Identity consists of following information

1. **A Key pair**: An Asymmetric RSA Key pair Associated with the user’s email address.
2. **Configuration**: Users configurations for Galaxkey clients.
3. **Users’ Password**: User’s password for authentication.

Although password is part of identity it is not stored with Galaxkey. User’s password is used as symmetric key for securing the rest of user’s identity. Thus, entering correct password gives access to the identity to the user and ONLY user will have access to it.

Thus, every user’s identity is secured using different symmetric key chosen and managed by end user making it very robust security model.

Section 5. Galaxkey and Entrust nShield HSM

5.1.Galaxkey and nShield

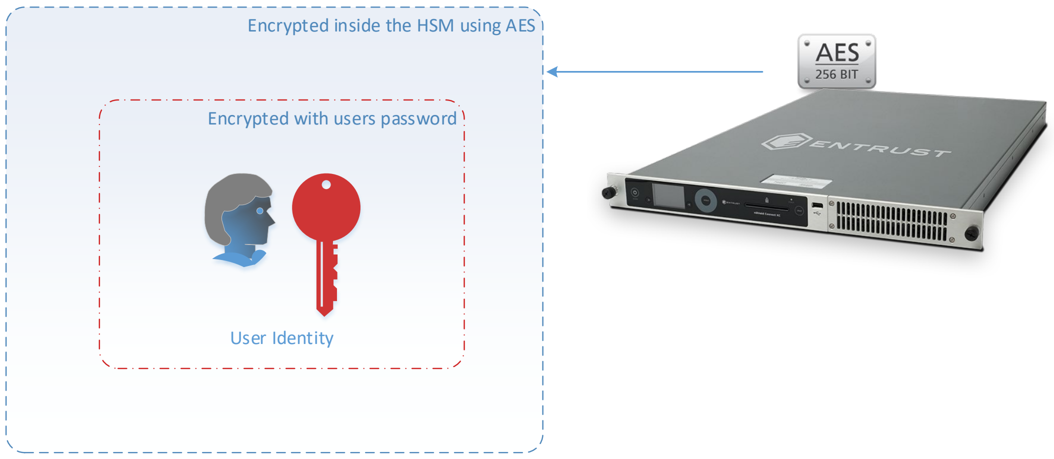
As seen from the architecture and security of Galaxkey, there is only one sensitive element in the system that can utilise the extra precaution of HSM security and that is user's private key.

Although the private key is stored in an encrypted format using users password it is desirable and at some places mandatory to have another unknown master key to secure the private keys of users. This is where the HSM provides the necessary security.

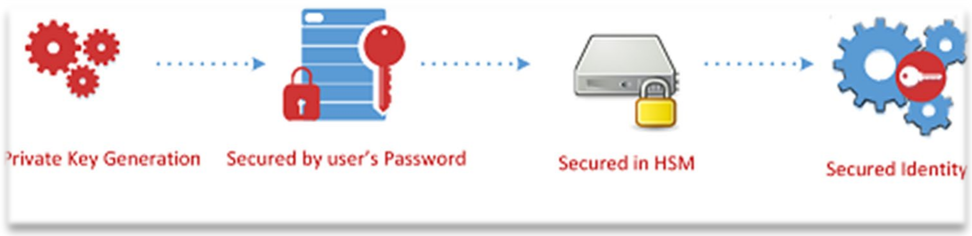
5.2. Galaxkey and nShield Integration

Galaxkey integrates with HSM in following three steps

1. **nShield Key generations**: For every key storing server (in cloud with Galaxkey or in-house with hybrid appliance) a nShield HSM is paired. A symmetric key is generated inside the HSM which will be used as master key for securing private keys on the server.


Encrypted inside the HSM using AES
Encrypted with users password
User Identity
AES
256 BIT
ENTRUST

2. **Securing Private Key**: Whenever a key is generated for a user, it is first secured using users' password and then secured using nShield HSM. The encryption will happen inside HSM and the secured private key will be stored on the server for further usage.



3. Getting Private Key: Whenever a client demands a private key, the key is loaded from storage into nShield HSM for decryption the decrypted key is then provided to the client. The client then decrypts it further using user's password and uses for decryption.

Secured Identity
Decrypted in HSM
Secured by user's Passwor

5.3.Integration Details

Galaxkey uses PKCS 11 standard protocol to integrate with nShield HSM. Following are specific code snippets for integrating with nShield

Integration is in 3 steps

1. **Step 1**: Initialization of nShield HSM: this function checks for a key with unique id in the HSM. If the key is not found a new key is generated. This happens only once for an installation.

// Login as normal user

```
session.Login(CKU.CKU_USER, Settings.NormalUserPin);

// look for AES 256 key with the specified Key ID
List<ObjectAttribute> objectAttributes1 = new List<ObjectAttribute>();
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_KEY_TYPE, CKK.CKK_AES));
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_ID, ckaId));
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_CLASS, CKO.CKO_SECRET_KEY));


List<ObjectHandle> foundObjects = session.FindAllObjects(objectAttributes1);
ObjectHandle Objhandle = null;
if (foundObjects.Count > 0)
{
       // if found, we are ok.
       Objhandle = foundObjects[0];
}
else
{
       // if not found, we need to create one, so set the attributes and generate it in
       HSM
       // Prepare attribute template of new key
       List<ObjectAttribute> objectAttributes = new List<ObjectAttribute>();
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_CLASS, CKO.CKO_SECRET_KEY));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_KEY_TYPE, CKK.CKK_AES));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_ENCRYPT, true));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_DECRYPT, true));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_DERIVE, true));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_EXTRACTABLE, true));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_TOKEN, true));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_ID, ckaId));
       objectAttributes.Add(new ObjectAttribute(CKA.CKA_VALUE_LEN, (ulong)32));


       // Specify key generation mechanism
       Mechanism mechanism = new Mechanism(CKM.CKM_AES_KEY_GEN);

       // Generate key
       ObjectHandle generatedKey = session.GenerateKey(mechanism, objectAttributes);
}
session.Logout();
```

2. **Step 2**: Encrypt the generated key using nShield HSM

Whenever a new key is generated, it is secured using user's password and it is further secured using AES key in nShield HSM. Following is the code snippet used for it.

```
session.Login(CKU.CKU_USER, Settings.NormalUserPin);

// look for the AES 256 key with the id that this instance was initialized with.
List<ObjectAttribute> objectAttributes1 = new List<ObjectAttribute>();
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_KEY_TYPE, CKK.CKK_AES));
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_ID, ckaId));
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_CLASS, CKO.CKO_SECRET_KEY));


List<ObjectHandle> foundObjects = session.FindAllObjects(objectAttributes1);
ObjectHandle Objhandle = null;
if (foundObjects.Count > 0)
{
    Objhandle = foundObjects[0];

    // get the IV in bytes
    byte[] iv = Encoding.UTF8.GetBytes(strIV);
    Mechanism mechanismEnc = new Mechanism(CKM.CKM_AES_CBC_PAD, iv);

    // source data in bytes
    byte[] sourceData = ConvertUtils.Utf8StringToBytes(strInput);

    // Encrypt data
    byte[] encryptedData = session.Encrypt(mechanismEnc, Objhandle, sourceData);

    // convert to base 64 for storing.
    strReturn = Convert.ToBase64String(encryptedData);
}
session.Logout();
```

3. **Step 3**: Decryption of the key using nShield HSM.

Whenever the key is demanded by the client it is first decrypted in the nShield HSM using the key generated in the nShield HSM. After decryption what we have is user's private key secured using his / her password. This private key is then passed on to client as is and client authenticates using the appropriate decryption method. Following is the code snippet for decrypting the key

```
// Login as normal user
session.Login(CKU.CKU_USER, Settings.NormalUserPin);

// look for the AES 256 key with the id that this instance was initialized with.
List<ObjectAttribute> objectAttributes1 = new List<ObjectAttribute>();
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_KEY_TYPE, CKK.CKK_AES));
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_ID, ckaId));
objectAttributes1.Add(new ObjectAttribute(CKA.CKA_CLASS, CKO.CKO_SECRET_KEY));
```

```
List<ObjectHandle> foundObjects = session.FindAllObjects(objectAttributes1);
ObjectHandle Objhandle = null;
if (foundObjects.Count > 0)
{
    Objhandle = foundObjects[0];
}
if (foundObjects.Count > 0)
{

    // get the IV in bytes
    byte[] iv = Encoding.UTF8.GetBytes(strIV);
    Mechanism mechanismEnc = new Mechanism(CKM.CKM_AES_CBC_PAD, iv);

    // source data in bytes from base 64 string.
    byte[] sourceData = ConvertUtils.Base64StringToBytes(strInput);

    // Decrypt data
    byte[] decryptedData = session.Decrypt(mechanismEnc, Objhandle, sourceData);

    // return data as string.
    strReturn = ConvertUtils.BytesToUtf8String(decryptedData);
}
session.Logout();
```

Section 6. Contact

In case there are any queries related to this document, or related to the product represented by the document, or if there is a query regarding Galaxkey, please contact us at *support@galaxkey.com*.