Modernisation, développement d'applications et DB2 sous IBM i

Technologies, outils et nouveautés 2012-2013

8 et 9 avril 2013 – IBM Client Center Paris, Bois-Colombes

S22 - Cryptographie

mardi 9 avril – 13h30-15h15

Habib SAAD – GAIA Mini Systèmes
jmsaad@gaia.fr

S22 - Cryptographie

PLANTONS LE DÉCORS

Cryptographie - définition

- **WIKIPEDIA** :
- La **cryptographie** est une des disciplines de la cryptologie s'attachant à protéger des messages (assurant confidentialité, authenticité et intégrité) en s'aidant souvent de ***secrets*** ou de ***clés***. Elle se distingue de la **stéganographie** qui fait passer inaperçu un message dans un autre message alors que la cryptographie rend un message inintelligible à une personne autre que qui-de-droit.
- En résumé : vous avez le message sous les yeux mais vous n'êtes pas capable de lire l'information qu'il contient sauf si vous êtes dans la confidence…

Cryptographie

- Message Digest / Hashage
  - Permet à partir d'un message M de créer facilement et rapidement un résumé unique R. Il s'agit d'une fonction à sens unique ne permettant pas, à partir du résumé R, de retrouver le message M.
    - Utilisation de MDx, SHAy.
  - Une modification du message modifie le Digest.
  - Cela ne garanti pas l'authenticité du message, seulement sa conformité à l'original.
- Signature numérique
  - Cryptage du message avec un système de clés asymétriques.
    - Lent
  - Cryptage du Digest avec un système de clés asymétriques.
    - Plus rapide du fait de la taille réduite du Digest.

Domaines d'utilisation

- **Transmission d'une information d'un point A à un point B**
  - **Seuls l'expéditeur et le destinataire doivent pouvoir lire l'information même si elle est interceptée**
    - **Messagerie (cryptage des bases courriers sous Lotus Notes)**
    - **Saisie de données personnelles dans un navigateur Web**
    - **Transmission de données bancaires, de fichiers, …**
- **Protection des éléments d'authentification**
  - **Protection du code secret d'une carte bancaire**
    - **Ce qui est protégé c'est le code qui est sur la carte**
  - **Badge d'accès (passage ou distributeur de boisson)**
    - **Cryptage des informations d'authentification**
- **Authentifier**
  - **Signature**

Domaines d'utilisation

- **Permet de protéger des données**
  - **Contre l'espionnage industriel**
  - **Contre le vol**
  - **Contre la contrefaçon, l'altération des informations**

- **Requis**
  - **Par la loi : une donnée inaltérable est une preuve d'authenticité**
  - **Par l'industrie bancaire**
  - **Par le bon sens…**

Opérations de base

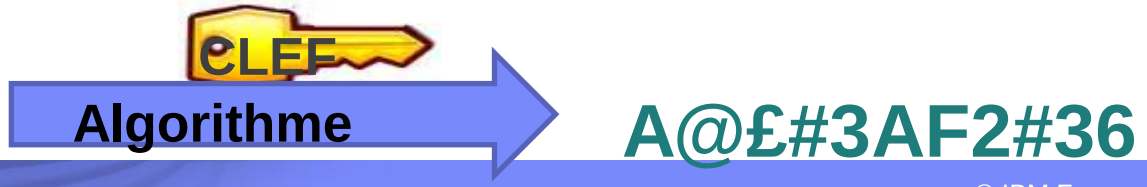
- **Brouillage des données en clair pour les rendre illisibles**
  - Encrypter ou crypter
- **opérations sur des données cryptées pour les obtenir en clair**
  - Décrypter
- C'est simple…comme un IBM i

Principe

- **Avoir un message (donnée) à encrypter (décrypter)**
- **Disposer d'un algorithme pour l'encryptage (décryptage)**
  - **Ensemble d'opérations effectuées sur un message "en clair" pour le rendre "moins clair"**
    - **Suites d'étapes finies non-ambigües permettant d'obtenir un résultat**
    - **Souvent des opérations mathématiques**
    - **Les algorithmes de cryptage sont peu nombreux et connus**

Mais la clé d'un cryptage efficace est…la clé !!

- **Disposer d'une chaine de caractères servant de clé**
  - **La qualité dépend de sa longueur et de son contenu**
  - **Elle dépend aussi des personnes la connaissant**
  - **Donc mieux vaut ne pas la connaître !**



Les différents algorithmes

Sur les IBM i nous avons (min. V5R4M0) :

ALGORITHME	NOM	TAILLE CLE (bits)
DES	Data Encryption Standard	56 (7 octets + 1 octet de parité)
3DES ou TDES	Triple DES	56, 112 ou 168 (7, 14 ou 21 octets)
AES	Advanced Encryption std	128, 192, 256 (16 octets -> 32 octets)
RC2	Rivest Cipher	8-1024
RC4-compatible	Rivest Cipher	8-2048
RSA	Rivest Shamir Adleman	512-2048
MD5-HMAC	Message Digest	HMAC – minimum 128
SHA1-HMAC		HMAC – minimum 160
Diffe-Hellman		512-1024

Algorithme de chiffrement par bloc (block-cypher alg.)

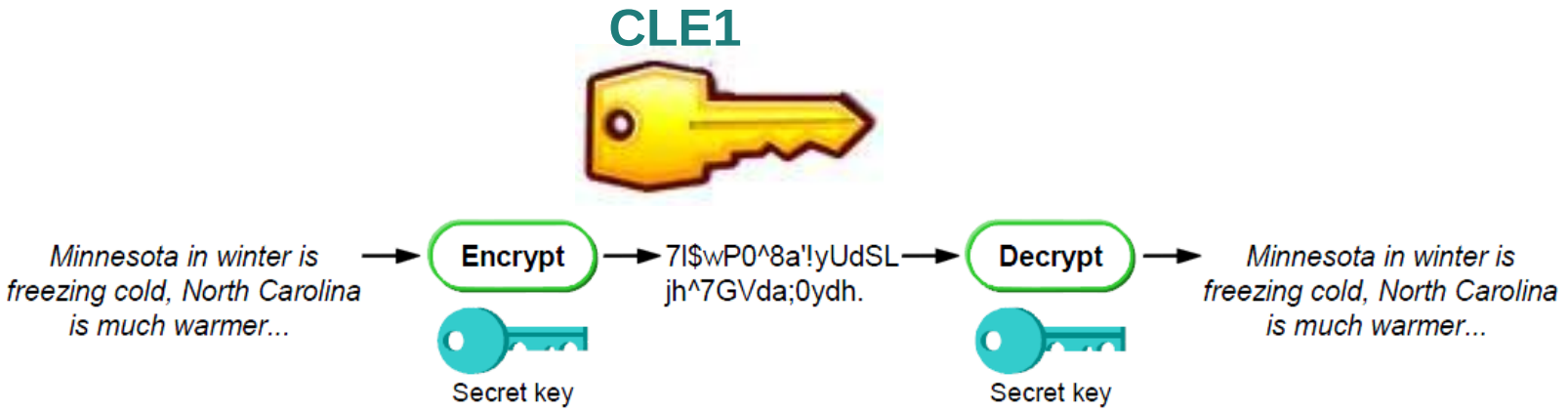
- **DES, 3DES, AES, RC2**
  - **Découpage des données en bloc de taille fixe (standard : 128 bits)**
  - **Les blocs sont encryptés les uns après les autres**
  - **Les données trop courtes sont complétées**
    - L'analyse des blocs permet d'identifier la taille réelle de la donnée en identifiant les caractères se répétant à la fin.
    - Sensible aux erreurs : erreur sur un bit ➔ erreur sur l'ensemble du cryptage.

Algorithme de chiffrement par flot (stream-cypher)

- **RC4-compatible (WEP)**
  - **Chiffrement de données de longueur quelconque**
  - **Aucun découpage nécessaire**
    - Chiffrage bit à bit avec une clé de même taille que le message
    - Inviolable et indépendance des bits entre eux

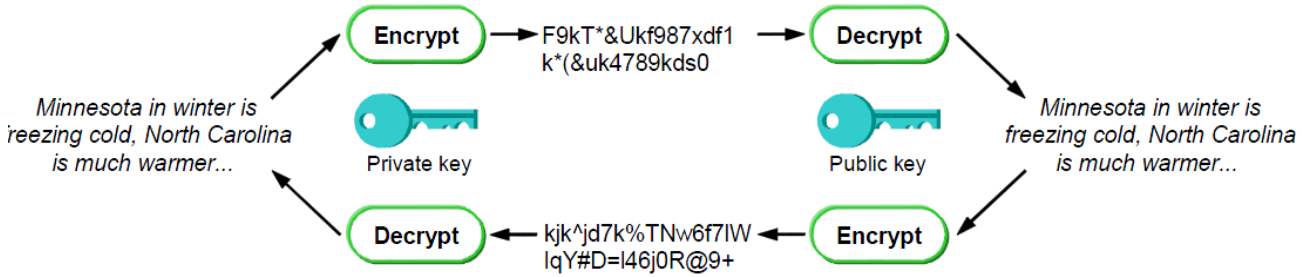
Cryptage à clé symétrique

- **La même clé sert au cryptage et au décryptage**
  - **Il est d'usage que ces types de clés ne circulent pas sur les réseaux et restent sur les serveurs**
  - **Il est aussi d'usage d'encrypter ces clés pour ne pas les avoir en clair.**
    - **On parle alors de Key-Encrypted-Key : clé encryptée par une autre clé (Key-Encrypting-Key) et via un algorithme de chiffrement.**



Cryptage à clé asymétrique

- **Une clé sert au cryptage**
  - **C'est une clé dite "publique" qui peut être transmise sans risque**
  - **Elle ne peut pas décrypter ce qu'elle a crypté**
- **Une clé sert au décryptage**
  - **C'est une clé dite "privée" qui ne doit jamais être transmise**
  - **Elle est conservée sur un serveur et/ou par l'expéditeur de la clé publique**
- **Elles fonctionnent par paire : publique/privée**



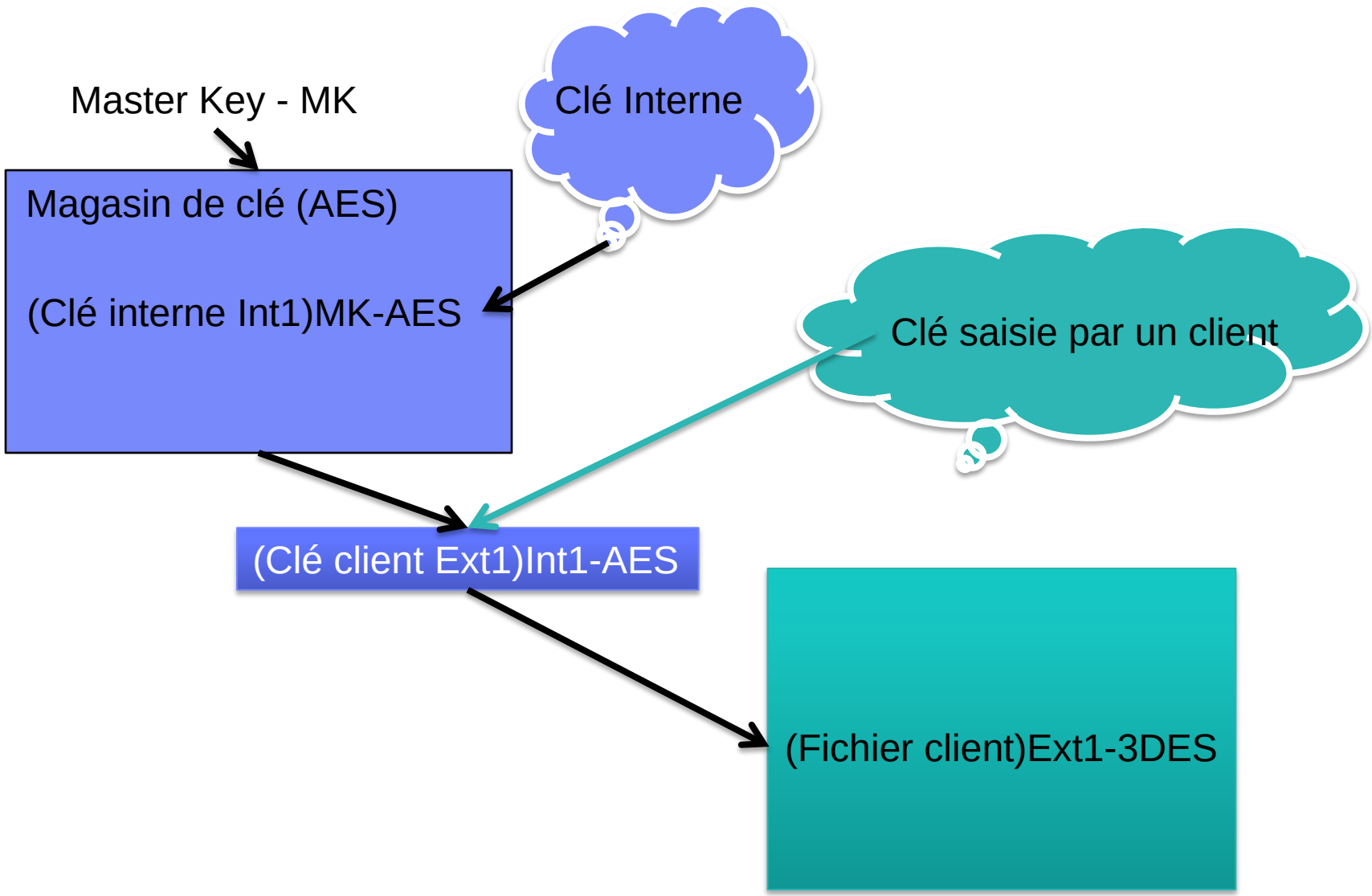
Stockage des clés

- En clair dans un fichier, une data Area, …
  - Dangereux
- Dans un fichier protégé
  - Un peu mieux
- Crypter la clé et la stocker dans un fichier
  - Solution sécurisée mais il faut une clé pour crypter la clé…
- Stocker la clé dans un magasin de clés
  - Une des meilleures solutions
- Stockage des clés sur des supports physiques
  - IBM Cryptographic hardware options 4764 et 4758
- Un mixte de plusieurs solutions
  - Au sommet de la pyramide une (ou plusieurs) clé(s) maîtresse(s)

Schéma de principe

- Une clé maîtresse ou principale (Master Key)
  - Propre à un système
  - Connue de peu de personnes (ou de personne)
  - Elle crypte les autres clés
  - Conservée en plus dans un coffre
- Une clé par application, par client
  - Saisie par le client, le gestionnaire d'applications
  - Connue de lui seul et sous sa responsabilité
  - Cryptée par la clé maîtresse ou par une autre clé…cryptée par la clé maîtresse
- Les clés sont stockées dans des magasins de clés
  - Cryptés par une Master Key
- Les clés peuvent utiliser des algorithmes différents

Schéma de principe : un exemple



S22 - Cryptographie

QUID DE L'AS400 ?

Le cryptage est ancien sur l'AS400 – S/38

- IBM System 38 dans les années 70.
- Produit sous licence 57xx-CR1
  - APIs et commandes CLP pour du cryptage/décryptage DES, de l'authentification de message (MAC : message-authentication code)
  - Interface Machine (MI) CIPHER en place depuis 1998 et utilisé jusqu'à la version V5R4.
    - DES
- Ajout de nouvelles interfaces depuis la V5R4
  - CCA (*Common Cryptographic Architecture)*
    - APIs
    - Matériel de cryptographie
      - Co-processeur de cryptographie IBM 2620 (déprécié), IBM 4758, IBM 4764, 4801
    - i5/OS 57xx-SS1 option 35 (CCA Cryptographic Service Provider [CCA CSP])

Le cryptage est ancien sur l'AS400 – S/38

- V5R2 :
  - Nouvelle architecture dans l'IBM Licensed Internal Code
- V5R3 :
  - APIs de cryptographie pour l'utilisation des CSP utilisant un accélérateur de cryptographie (4805 – 2058) ou au travers du LIC.
    - Nécessite Cryptographic Access Provider product (57xx-AC3).
  - Fonctions de cryptographie en SQL (ENCRPYT_RC2)
- V5R4 :
  - APIs de cryptographie en standard,
  - Les produits 57xx-AC1 and 57xx-AC2 deviennent obsolètes
  - Fonctions de cryptographie en SQL (ENCRPYT_TDES)
- I6.1 :
  - fonction SQL : ENCRYPT_AES

Les APIs sont rangées sous six catégories

- APIs de cryptage/décryptage
- APIs d'authentification
- APIs de génération de clés
- APIs PRNG – génération pseudo-aléatoire de nombres et algorithmes de génération de clés
- APIs de contexte Cryptographique
- APIs de gestion de clés

Plusieurs solutions s'offrent à nous sur l'IBM i

- Cryptage matériel
  - Carte accélératrice de cryptographie 2508 par exemple

(AS400 IBM 9406 IOP Controller, #4805 PCI Crypto Accelerator 8xx/)

- Cryptage logiciel
  - APIs de cryptographie
- En standard, IBM fourni les APIs donc… oublions les cartes de cryptographie pour cette fois.

Juste une petite comparaison entre les solutions

- Carte 2058 (CCIN) – accélérateur cryptographique PCI

Fonction : Qc3EncryptData/Qc3DecryptData	I5/OS	2058
DES ECB /CBC (Cypher Block Chaining)	✓	✓
DES OFB	✓	✗
DES CFB 1-bit /8-bit/64-bit	✓	✗
TDES ECB /CBC	✓	✓
TDES OFB	✓	✗
TDES CFB 1-bit/8-bit/64-bit	✓	✗
AES ECB / CBC	✓	✗
RC4	✓	✗
RSA	✓	✗

Clé maîtresse ou principale – Master Keys

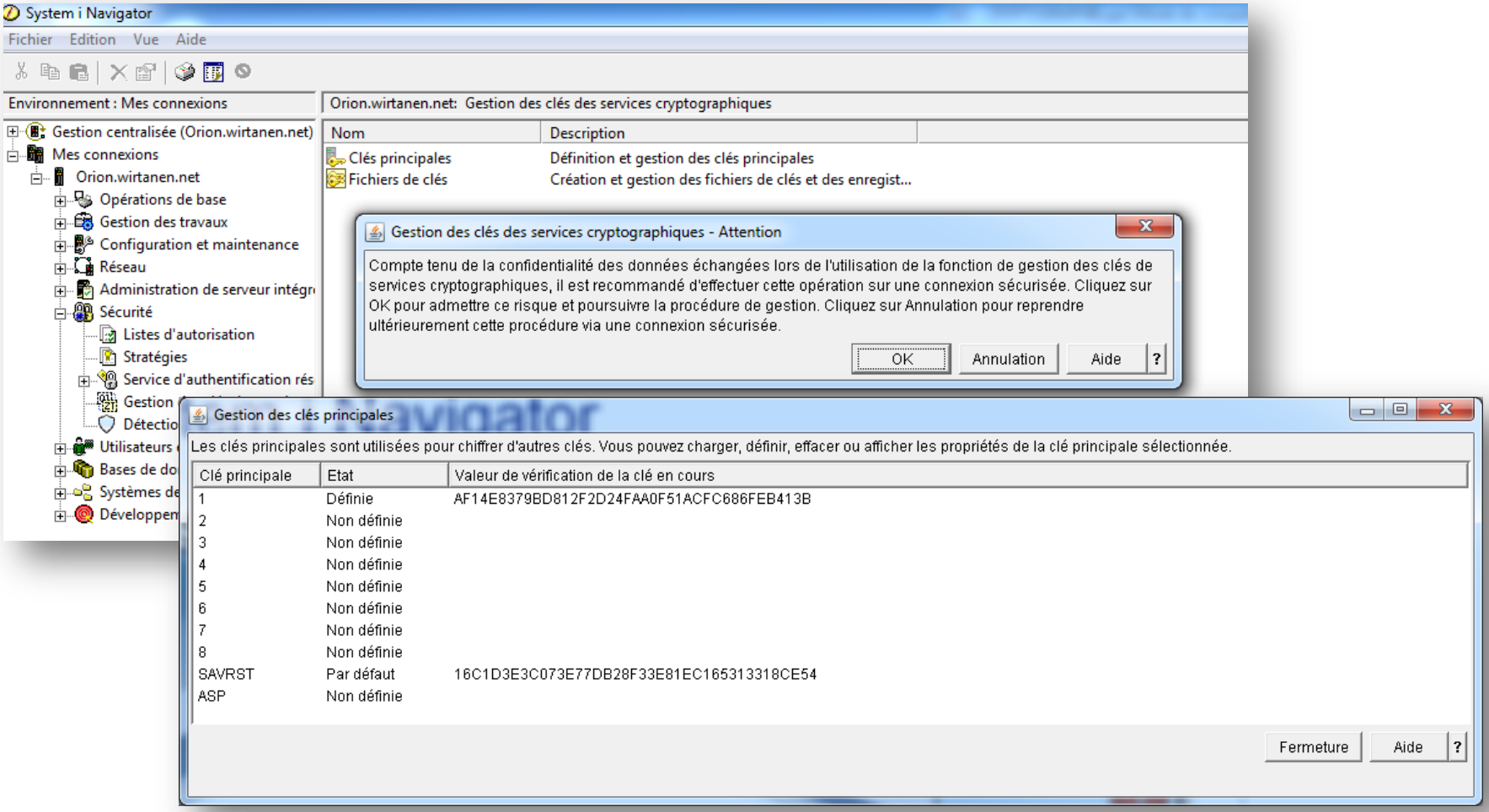
- Il s'agit d'une clé liée à une machine
  - 32 octets (256 bits) AES
- Elle est saisie une fois et permet de crypter d'autres clés
  - Impossible de la dupliquer
  - Impossible de crypter autre chose que des clés ou des magasins de clés
  - Non récupérable
  - Sauvegardée par un SAVSYS uniquement
- V5R3 : une seule clé maîtresse
- A partir de la V5R4 : jusqu'à 8 clés maîtresses
  - Cela rend plus pratique le changement régulier des clés de cryptage

Master Key

- Composée de plusieurs parties (ordre de saisie non pris en compte)
  - Stockée par la machine
  - Chaque partie est chargée (load)
  - La clé principale est ensuite créée (set) et retourne une donnée de validation (Validation Key)
  - Elle est versionnée : OLD (celle remplacée), NEW (celle chargée) et CURRENT (la master key en cours).
  - Elle ne peut pas être affichée mais elle peut être mise à blanc
  - Elle peut être vérifiée grâce à sa valeur de validation de clé (KVV)
  - Sauvegardée par un SAVSYS

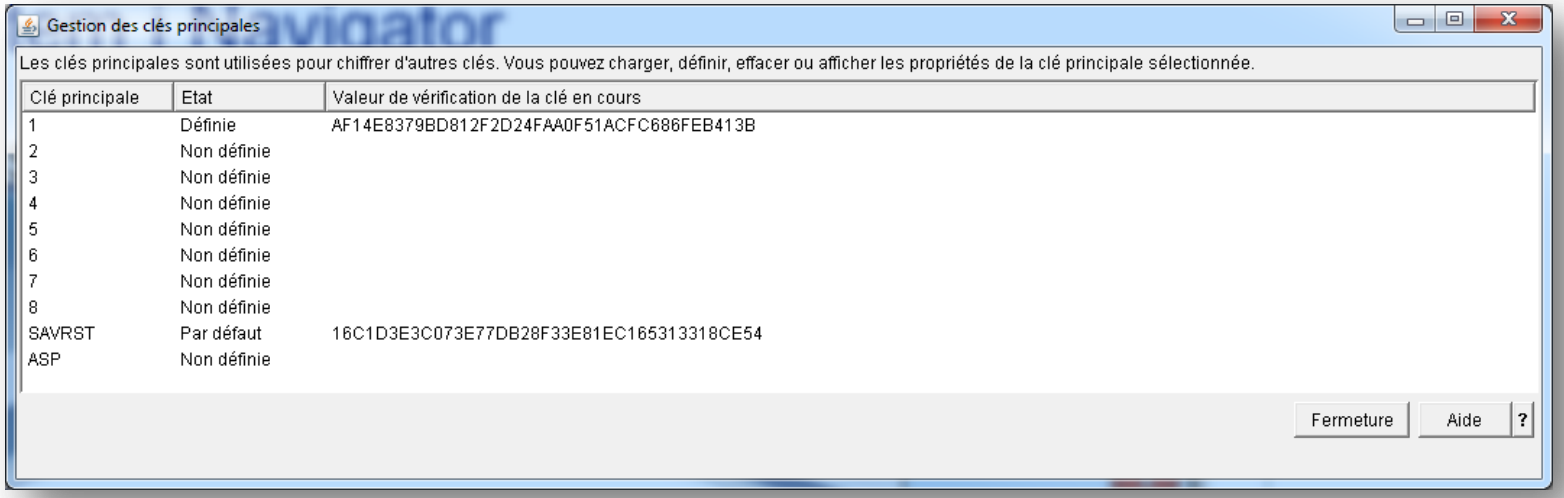
Via System i Navigator

- "votre-système" → Sécurité → Gestion des clés des services



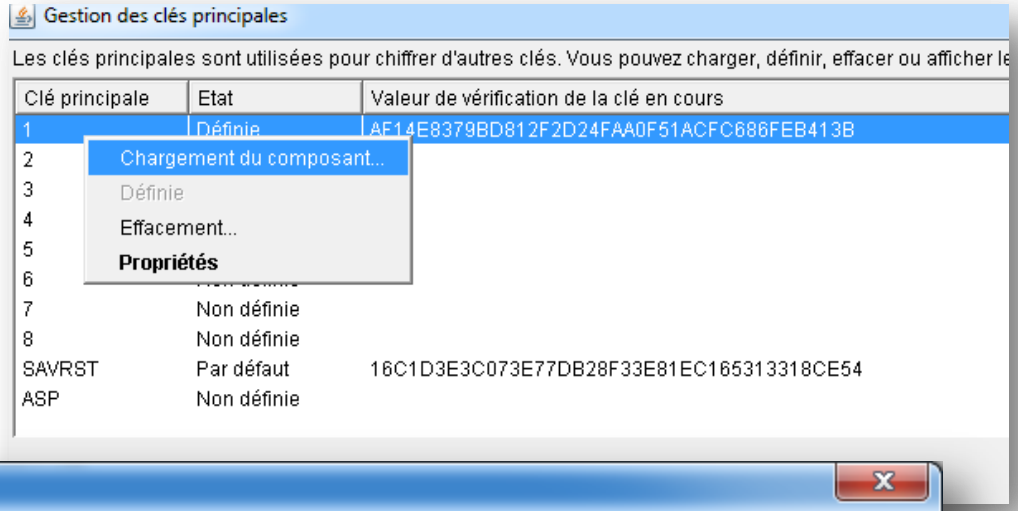
Via System i Navigator

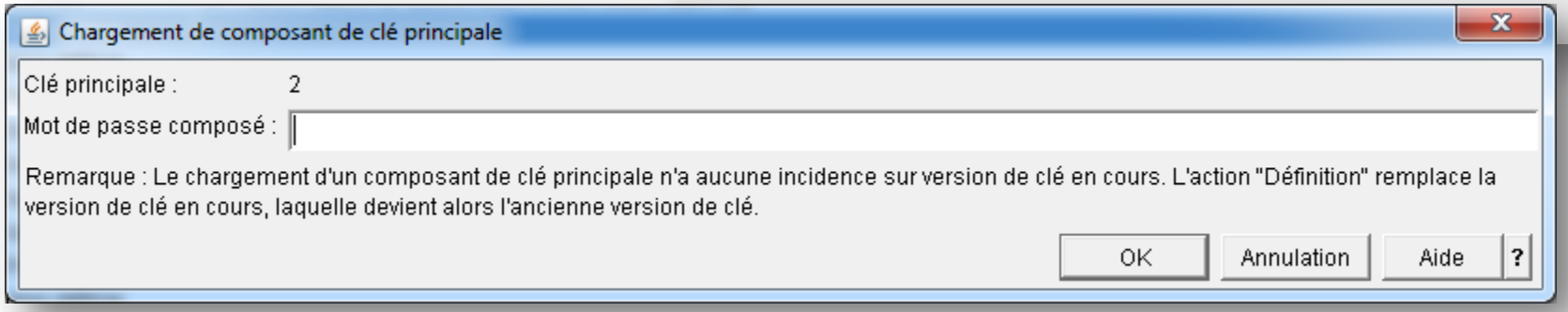
- Cette interface permet
  - de voir les 8 clés maîtresses (Master Keys)
  - Et une clé (SAVRST) qui permettra de sauvegarder les clés maîtresses via un SAVSYS



System i Navigator : chargement (load)

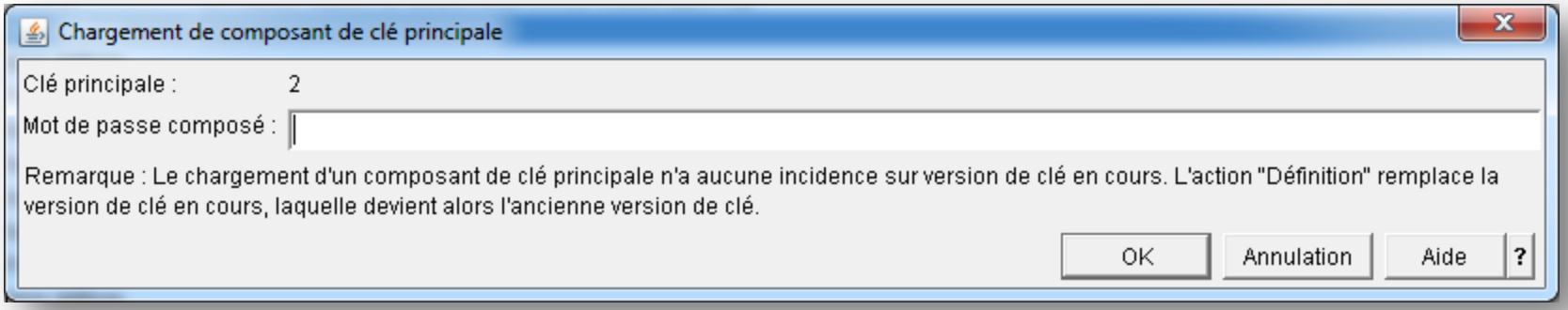
- Sur une clé existante ou une nouvelle Master Key
  - La clé actuelle deviendra la version OLD après affectation/définition (set).
  - Saisie en mode hexadécimal ou non
  - 256 bits (32 octets) AES





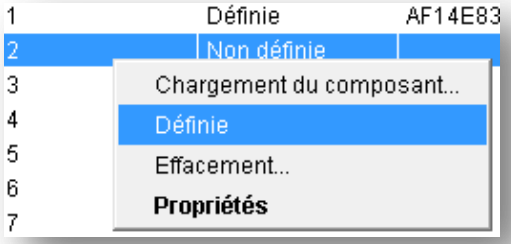
System i Navigator : chargement (load)

- L'ordre a peut d'importance
- Il vaut mieux la scinder en plusieurs parties détenues chacune par une personne
- Le chargement n'influe par sur une éventuelle clé existante
- Elle n'est activée qu'après affectation/définition (set)
  - Pour l'instant ce n'est que la version NEW

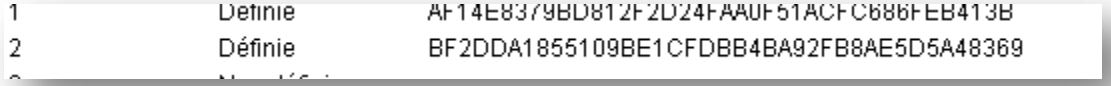


System i Navigator : définition (set)

- Clic-droit sur la clé chargée : définie
  - La clé précédente devient la version OLD (toujours utilisable)
  - La nouvelle devient la version CURRENT

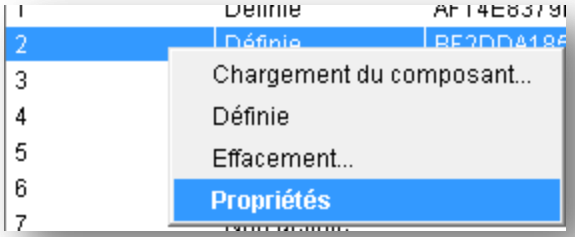


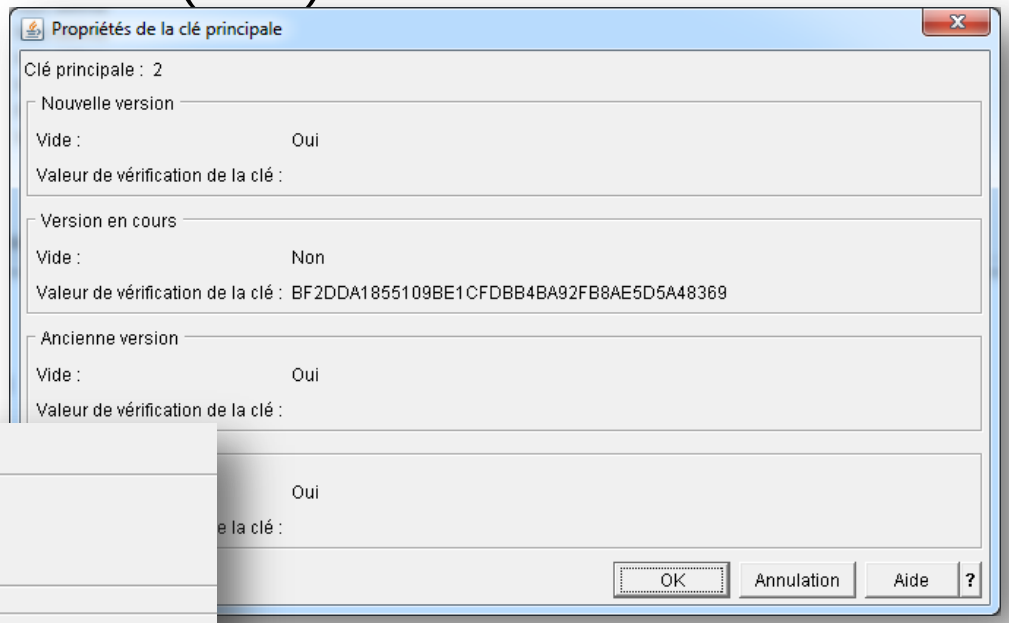
- Une valeur de validation de clé (KVV) est générée
  - Unique, 10 caractères (20 en hexadécimal)
  - Elle devra être stockée comme les différentes parties pour tester la clé sans la ressaisir



System i Navigator : propriétés

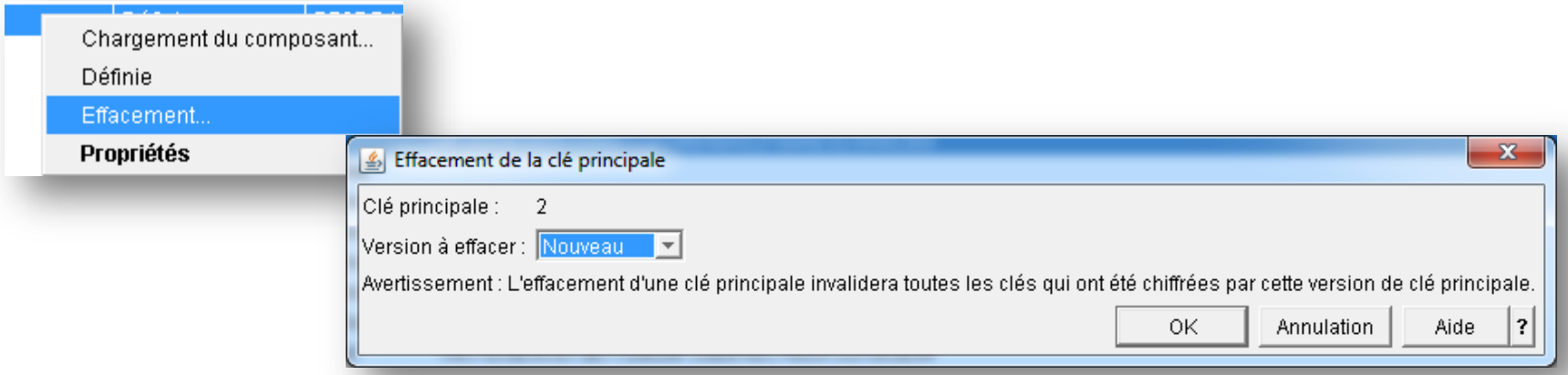
- Permettent de vérifier si
  - Une nouvelle version est chargée (NEW) avec sa KVV temporaire
  - Une version est en cours (CURRENT) avec sa KVV
  - Une ancienne version est présente (OLD) avec sa KVV





System i Navigator : effacement d'une Master Key

- Effacement
  - de la clé en cours
  - Effacement des composants chargés
- Tous les éléments cryptés avec la clé effacée sont perdus !!
  - Soit on a pensé à tout
  - Soit Pôle Emploi vous proposera un rendez-vous



Conclusion System i Navigator

- Simple
- Clair
- Mais
  - Trop standard : les composants saisis donneront la clé
  - Pas assez sécurisé

- Pour plus de sécurité, préférez les APIs.
  - Attention aux droits !!! L'adoption de droits ne marche pas !!!

Utilisation des APIs

- En standard avec tous les System i depuis la V5R3
  - ILE via des programmes de service
  - Équivalent OPM pour les allergiques à l'ILE
- Remplace le produit (plus fourni depuis la V5R4)
  - Cryptographic Support for AS/400® 5722-CR1.
    - Seulement du DES/3DES ?

Chargement des composants – Load Master Key Parts

- API QC3LDMKP ou Qc3LoadMasterKeyPart
  - Programme de service **QC3MKPLD (*EXCLUDE)**
  - Permet de saisir de 1 à n parties de la clé sous forme de phrase de chiffrement (passphrase) en hexadécimal ou non
  - L'ordre de saisie n'a pas d'importance
  - Chaque partie ne doit être connue que d'une personne

Description	Utilisation	Type
ID clé maîtresse	Entrée	Binaire(4)
Phrase de chiffrement (passphrase)	Entrée	Char(*)
Longueur de la phrase de chiffrement	Entrée	Binaire(4)
CCSID de la phrase de chiffrement	Entrée	Binaire(4)
Code erreur	E/S	Char(*)

Chargement des composants – Load Master Key Parts

- En passant par des prototypes

```
  // 0 : CCSID du travail sinon 1 à 65533
 D JOB_CCSID       c                   0
  **-- Load master key part:
 D LoadMasterKey   Pr                  ExtProc( 'Qc3LoadMasterKeyPart' )
 D  MstKeyID                    10i 0 Const
 D  PassPhrase                 256a   Const  Options( *VarSize )
 D  PassPhrLen                  10i 0 Const
 D  PassPhrCcsId                10i 0 Const
 D  Error                    32767a          Options( *VarSize )
  **-- API error data structure:
 D ERRC0100        Ds                  Qualified
 D  BytPrv                      10i 0 Inz( %Size( ERRC0100 ))
 D  BytAvl                      10i 0
 D  MsgId                        7a
 D                               1a
 D  MsgDta                     512a
…
 LoadMasterKey( PxMstKeyID
                : PxPassPhrase
                : %Len( PxPassPhrase )
                : JOB_CCSID
                : ERRC0100
                );
 // ERRC0100.BytAvl > 0 ==> Erreur
```

Affectation – Set Master Key

- API QC3SETMK ou Qc3SetMasterKey
  - Programme de service **QC3MKSET (*EXCLUDE)**
  - La clé chargée correspond à la version 'NEW'
  - Elle n'est pas encore affectée aux master Keys
  - Utiliser l'API Qc3SetMasterKey (ILE) ou QC3SETMK (OPM)

Description	Utilisation	Type
ID clé maîtresse	Entrée	Binaire(4)
Valeur de vérification KVV	Sortie	Char(20)
Code erreur	E/S	Char(*)

Chargement des composants – Load Master Key Parts

- En passant par des prototypes

```
 D SetMasterKey    Pr                  ExtProc( 'Qc3SetMasterKey' )
 D  MstKeyID                    10i 0 Const
 D  KeyVfyVal                   20a
 D  Error                    32767a          Options( *VarSize )
**-- API error data structure:
 D ERRC0100        Ds                  Qualified
 D  BytPrv                      10i 0 Inz( %Size( ERRC0100 ))
 D  BytAvl                      10i 0
 D  MsgId                        7a
 D                               1a
 D  MsgDta                     512a
…
 SetMasterKey( PxMstKeyID: KeyVfyVal: ERRC0100 );
 // ERRC0100.BytAvl > 0 ==> Erreur
```

  - La nouvelle clé devient la clé courante (CURRENT)
  - L'ancienne clé (si elle existe) devient la version OLD
  - Les deux coexistent mais seule la clé courante est utilisée
  - La migration des éléments cryptés par la version OLD vers la version CURRENT est prévue

Test – Test Master Key

- API QC3TSTMK ou Qc3TestMasterKey
  - Programme de service **QC3MKTST (*EXCLUDE)**
  - Permet de tester la clé maîtresse en retournant la valeur de vérification de clé (KVV)
  - Inutile de saisir la clé
  - Mais il faut indiquer la version

Description	Utilisation	Type
ID clé maîtresse	Entrée	Binaire(4)
Version de la clé maîtresse	Entrée	Char(1)
Valeur de vérification KVV	Sortie	Char(20)
Code erreur	E/S	Char(*)

Conclusion APIs

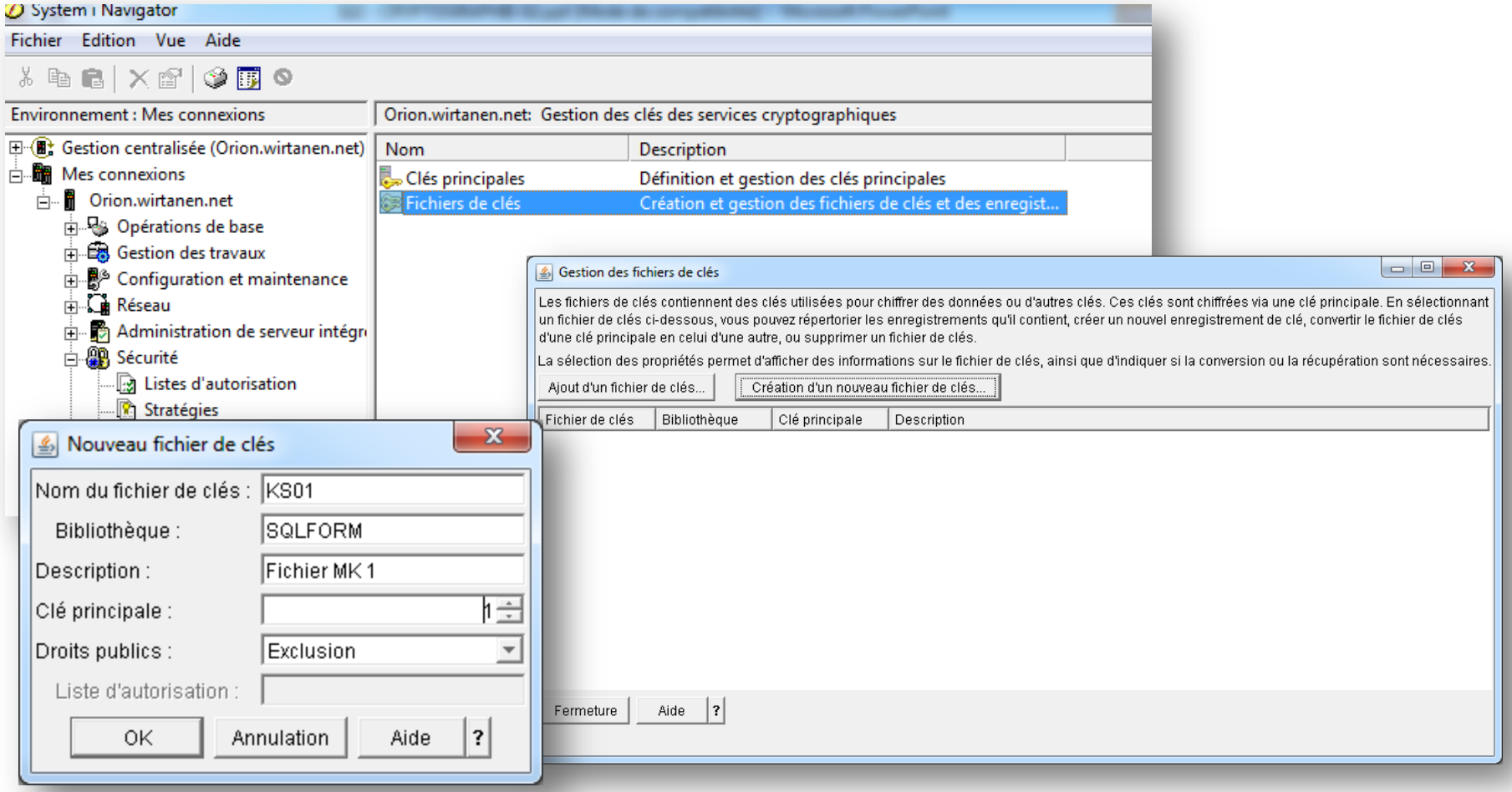
- Plus compliqué que System i Navigator, mais…
  - Plus de possibilités de gérer des clés…connues de personne !!
    - En effectuant des saisies de différents composants
    - En effectuant des opérations "maison" sur les composants
    - En proposant des systèmes de génération de composants aléatoires
    - …
- Dans tous les cas, les clés maîtresses ne servent qu'à crypter d'autres clés
  - En passant par des magasins de clés (Keystore)

Magasin de clés - Keystore

- Il est basé sur l'une des 8 Master Key de l'IBM i
- Il sert à stocker des clés en les cryptant avec la Master Key correspondante
- Peut-être utilisé par les APIs de cryptage, via des commandes ou via System i Navigator
- Chaque donnée cryptée est identifiée par un label de 32 caractères

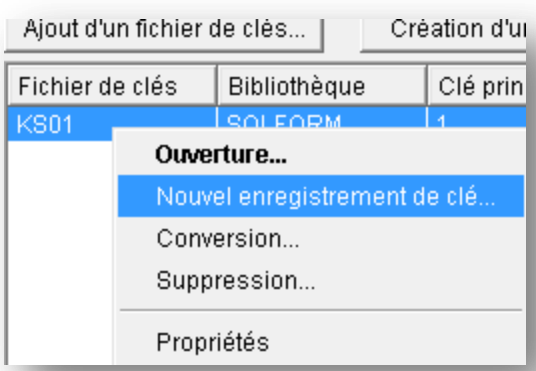
Via System i Navigator

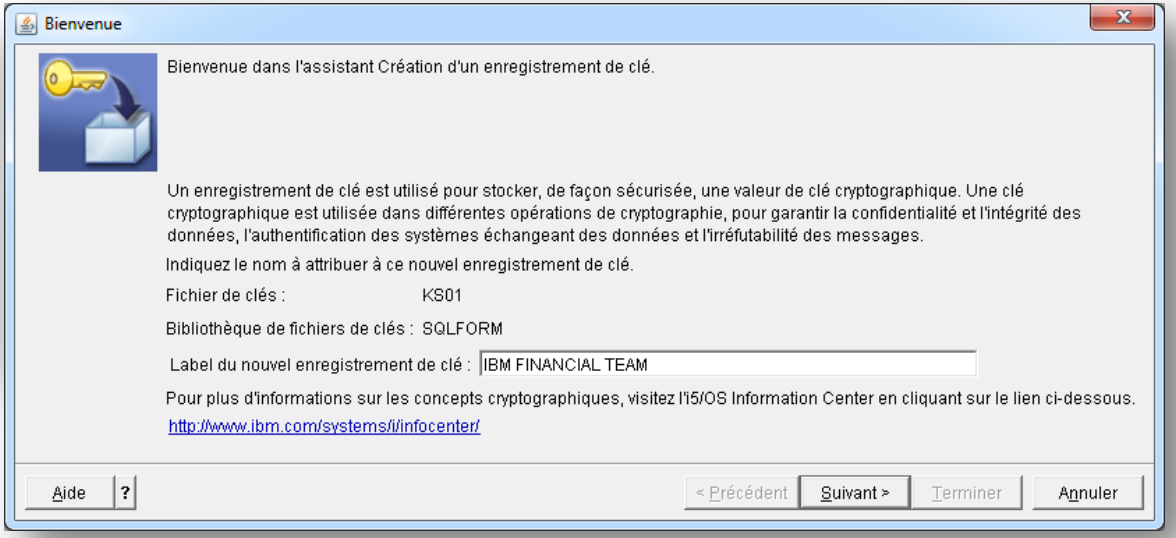
- "Votre-système" → Gestion des clés des services → Fichiers de clés



System i Navigator : ajout d'une clé à un magasin

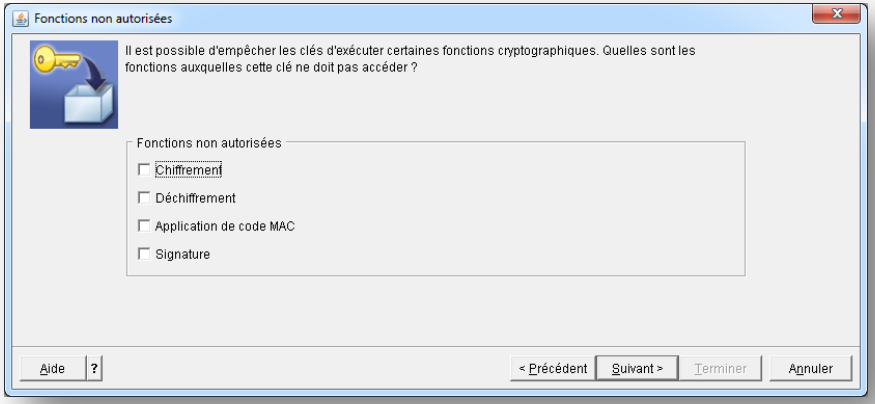
- Le keystore est crypté via une master Key en AES
- Les données qu'il contient sont donc illisibles
  - Mais elles sont identifiées par une étiquette (label)



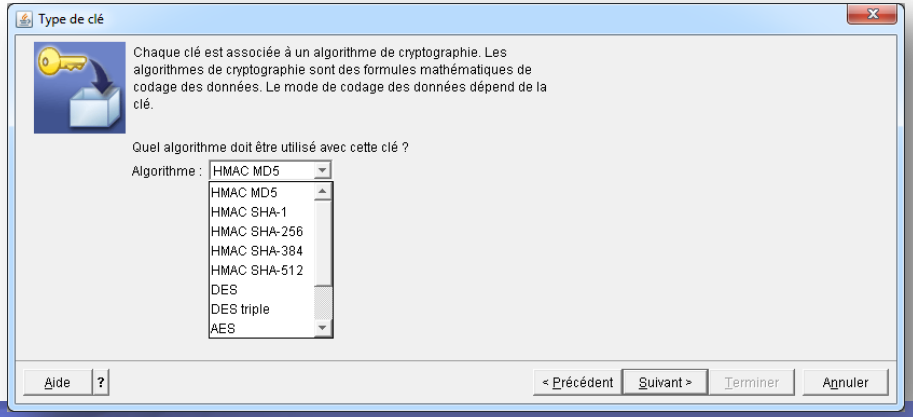


System i Navigator : ajout d'une clé à un magasin

- Les possibilités de la clé peuvent être limitées
  - Cocher ce qu'elle ne PEUT PAS faire
    - Chiffrer, déchiffrer, application du code MAC, signature

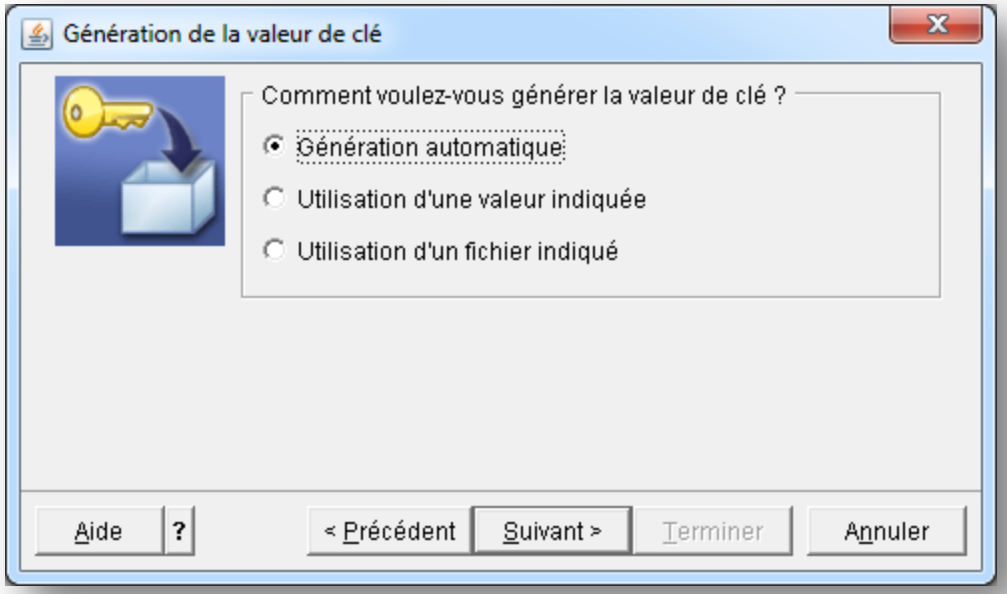


- Une clé est adaptée à un type de cryptage



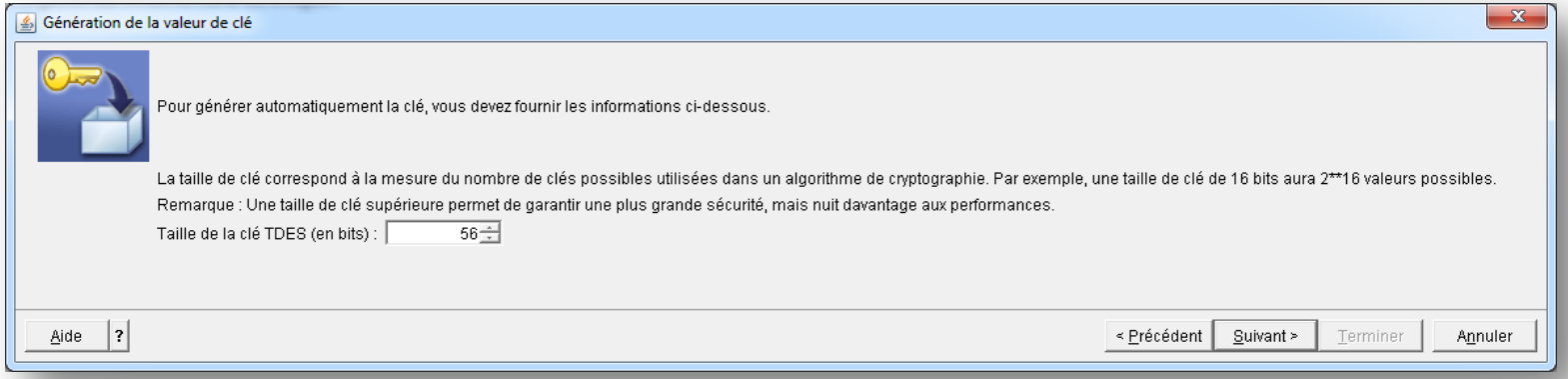
System i Navigator : ajout d'une clé à un magasin

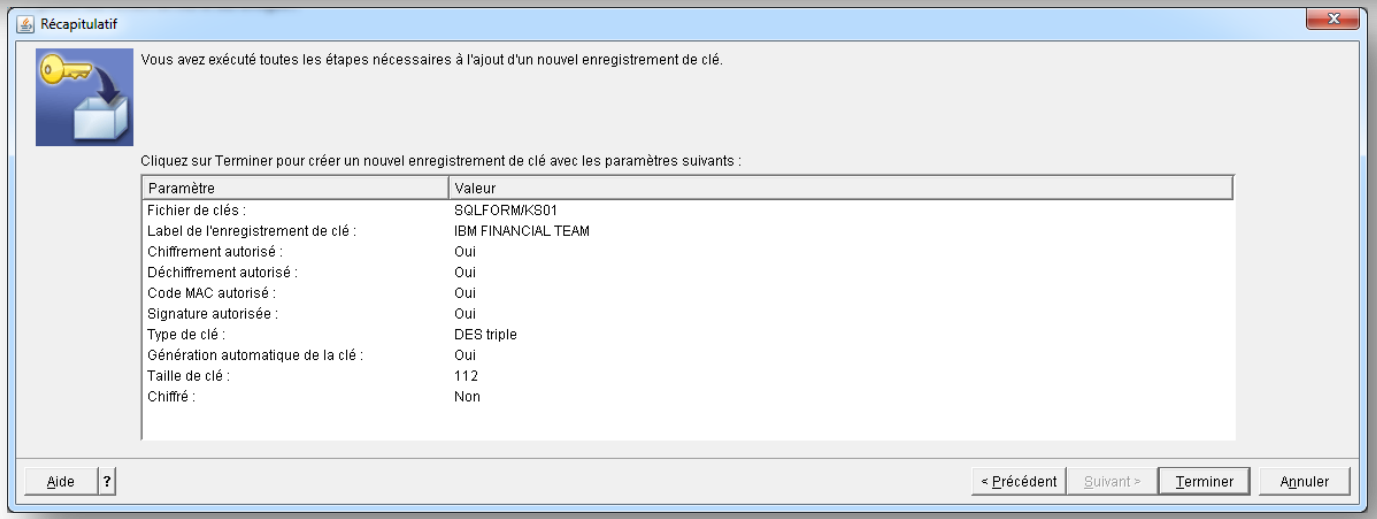
- Valeur de la clé
  - Auto-générée
  - Saisie
  - Récupérée d'un fichier



System i Navigator : ajout d'une clé à un magasin

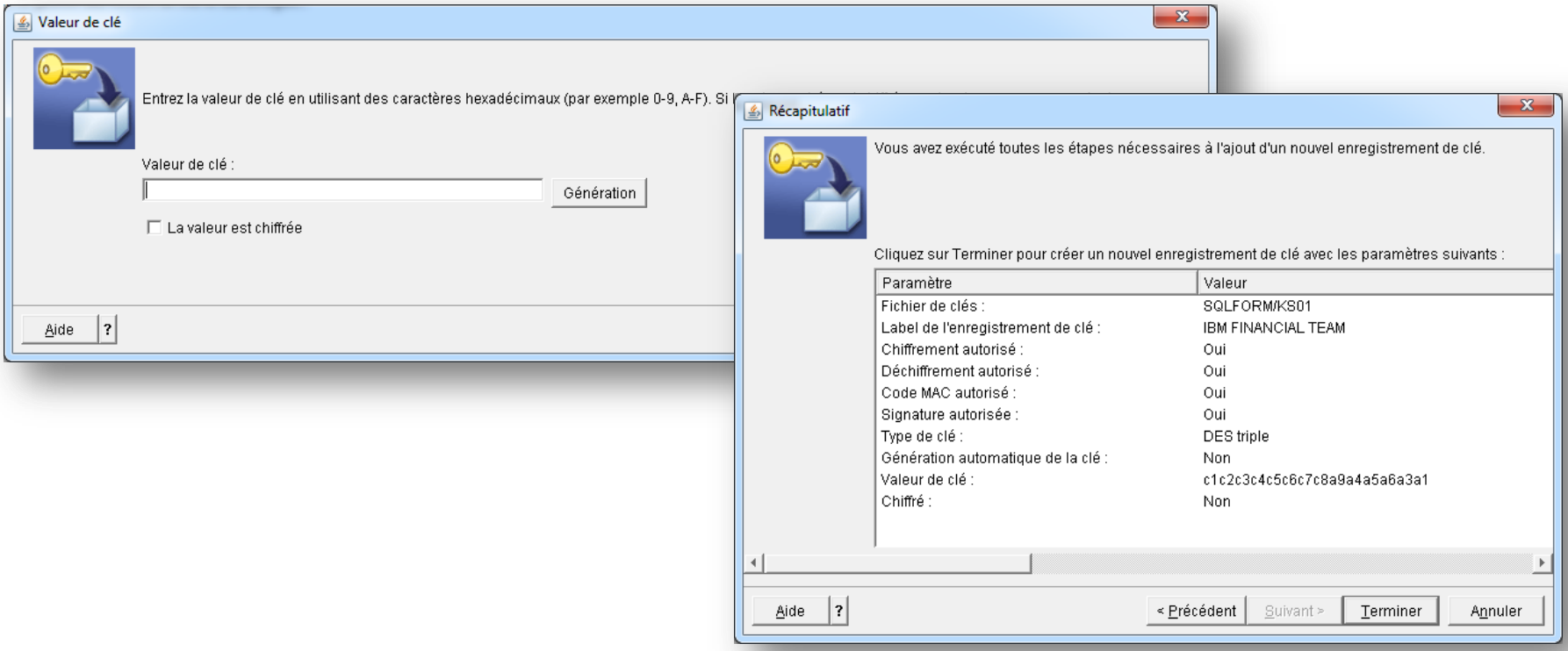
- Auto-générée
  - Indiquer la taille de la clé (fonction de l'algorithme)





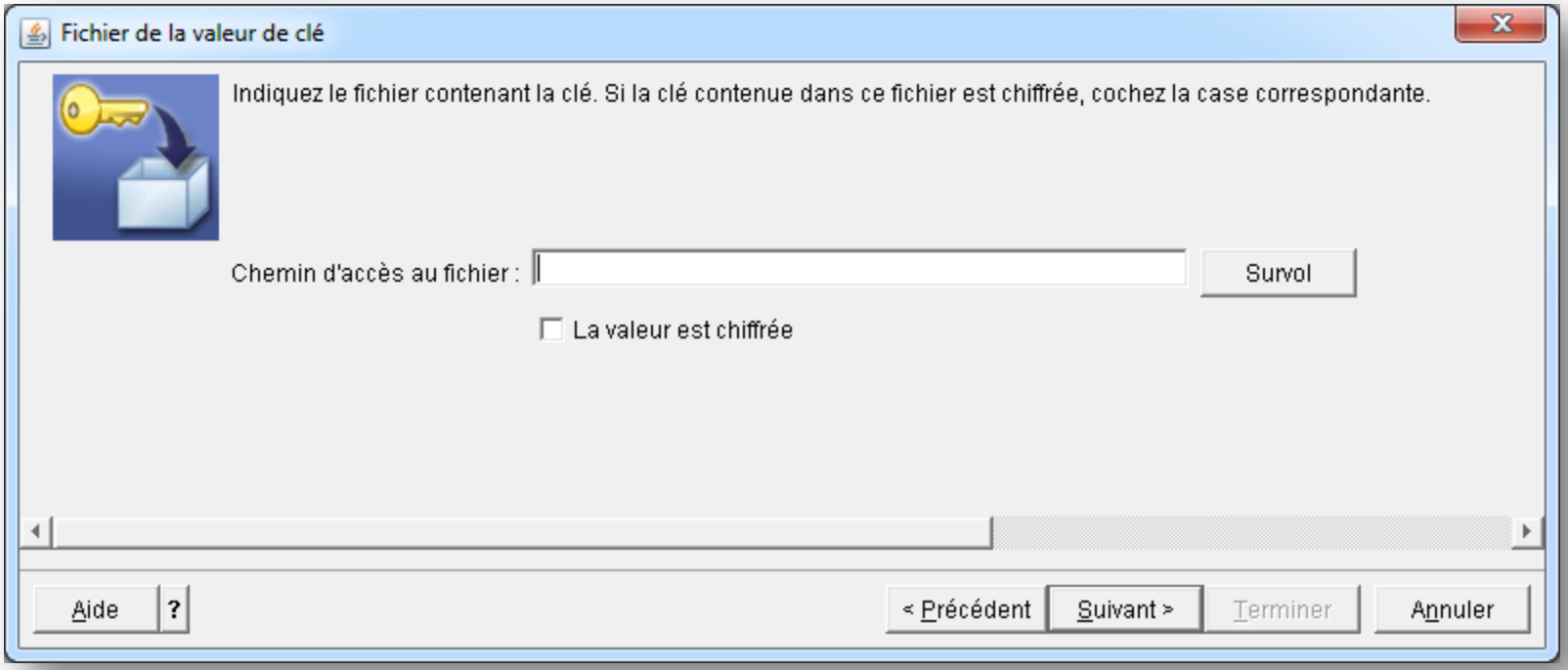
System i Navigator : ajout d'une clé à un magasin

- Valeur saisie
  - Tapez sa valeur (en hexadécimal)
  - Elle peut être en clair ou chiffrée
    - Si chiffrée, indiquer le fichier contenant la clé de décryptage



System i Navigator : ajout d'une clé à un magasin

- Valeur depuis un fichier
  - Indiquez le fichier contenant la clé
  - Elle peut aussi être cryptée
    - Dans ce cas, donner ensuite le fichier contenant la clé de décryptage



Via les APIs – création du KeyStore

- API QC3CRTKS ou Qc3CreateKeyStore
  - Programme de service **QC3KSCRT (*USE)**

Description	Utilisation	Type
Nom qualifié du magasin	Entrée	Char(20)
Identifiant clé maîtresse	Entrée	Binaire(4)
Autorisations publiques	Entrée	Char(10)
Texte descriptif	Entrée	Char(50)
Code erreur	E/S	Char(*)

Création du KeyStore

- En utilisant les prototypes

```
**-- Create KeyStore API :
D CreateKeyStore  Pr            extProc('Qc3CreateKeyStore')
D  KSQualName              20a   const
D  MstKeyID                10i 0 const
D  PublicAuth              10a   const
D  TextDesc                50a   const
D  Error                32767a   Options( *VarSize )
…
Callp CreateKeyStore(KSQualName
                     : MstKeyID
                     : AUT_EXCLUDE
                     : description
                     : ERRC0100
                     );
    If  ERRC0100.BytAvl > *Zero;
…
```

Structure du KeyStore

- Le fichier KeyStore à la structure suivante :

Nom	Type	Description
KYLABEL	Char(97)	Etiquette
RESRV1	Char(3)	Réservé
KEYTYPE	Binaire(4)	Type de clé
KEYSIZE	Binaire(4)	Taille de la clé
TIMEDATE	Char(4)	
KVV	Char(20)	Validation de la valeur de la clé
KYVARIANT	Binaire(4)	
CHECKSUM	Binaire(4)	Contrôle d'intégrité
RESRV2	Char(2)	Réservé
TOKEN	varchar(2414)	Jeton (longueur allouée : 32)

Opérations sur le KeyStore : stockage d'une clé

- API QC3WRTKR ou Qc3WriteKeyRecord
  - Programme de service **QC3KRWRT (*USE)**

Description	Utilisation	Type
Nom qualifié du magasin (fichier (10) biblio (10))	Entrée	Char(20)
Label	Entrée	char(32)
Clé	Entrée	Char(*)
Taille de la clé	Entrée	Binary(4)
Format de la clé	Entrée	Char(1)
Type de clé	Entrée	Binary(4)
Fonctions non autorisées	Entrée	Binary(4)
Format de clé (cryptée ou en clair)	Entrée	Char(1)
Jeton de clé de cryptage de la clé	Entrée	char(8)
Jeton d'algorithme de cryptage de la clé	Entrée	char(8)
Code erreur	E/S	Char(*)

Opérations sur le KeyStore : stockage d'une clé

- Nom qualifié
  - Fichier (char(10)) bibliothèque (char(10))
- Label
  - Nom unique désignant la clé
- Pour le reste, ça devient plus compliqué !!
- Format de (la chaine contenant) la clé :

Type	valeur
Chaîne binaire	0
Chaine BER (pour RSA)	1
Certificat PEM	6

Opérations sur le KeyStore : stockage d'une clé

- ça devient encore plus compliqué !!
- Type de clé :

Type	valeur
MD5	1
SHA-1	2
SHA-256	3
SHA-384	4
SHA-512	5
DES	20
Triple DES	21
AES	22
RC2	23
RC4	30
RSA et RSA privé	50 et 51

Fonctions exclues :

Type	valeur
Aucune	0
Cryptage	1
Décryptage	2
Mac	4
Signature	8

Format de la clé :

Type	valeur
En clair	0
Cryptée	1

Opérations sur le KeyStore : stockage d'une clé

- Prototype :

```
**-- Create key record in a Key Store  API
 D WrtKeyRcd       Pr                   ExtProc( 'Qc3WriteKeyRecord')
 D  KSQualName                   20a    Const
 D  RcdLbl                       32a    Const
 D  KeyStr                       32a    Const
 D  LenKeyStr                    10i 0  Const
 D  KeyFormat                     1a    Const
 D  KeyType                      10i 0  Const
 D  DisAlwFnc                    10i 0  Const
 D  KeyForm                       1a    Const
 D  KEKCtxTkn                     8a    Const
 D  KEKAlgTkn                     8a    Const
 D  Error                    32767a          Options( *VarSize )
```

- Mais comme ce sont souvent les même algorithmes, on peut simplifier

Opérations sur le KeyStore : stockage d'une clé

- En ne transmettant que les paramètres variables :

```
P WrtKRecord      B                   export
D                 PI
D KSQualName                    20a   const
D KeyStr                        32a   const
D KeKCtxTkn                      8a   const
D AlgCtxTkn                      8a   const
D Label                            32a
D KeyFormat       S              1a   inz('0')                          | binary key    |
D KeyForm         S              1a   inz('0')                          | key in clear  |
D LenKeyStr       S             10i 0 inz(32)
 /free
       Callp WrtKeyRcd( KSQualName
                      : label
                      : KeyStr
                      : LenKeyStr
                      : KeyFormat
                      : AES
                      : ALL_FCN
                      : keyForm
                      : kEKCtxTkn
                      : AlgCtxTkn
                      : ERRC0100
                      );

   If  ERRC0100.BytAvl > *Zero;
```

Opérations sur le KeyStore : stockage d'une clé

- Deux données paraissent encore obscures :
  - Les jetons de la clé de cryptage et de l'algorithme de cryptage

```
P WrtKRecord      B                   export
D                 PI
D KSQualName                    20a   const
D KeyStr                        32a   const
D KeKCtxTkn                      8a   const
D AlgCtxTkn                      8a   const
D Label                            32a
```

- Si la clé n'est pas cryptée, les deux paramètres
  - sont à blanc

    ou
  - ont un pointeur nul
- Si la clé est cryptée, alors
  - Jeton (valeur ou pointeur) de la clé de cryptage
  - Jeton (valeur ou pointeur) de l'algorithme de cryptage

Opérations sur le KeyStore : création des jetons

- API Qc3CreateAlgorithmContext ou QC3CRTAX
  - Programme de service **QC3CTX (*USE)**
  - Permet de créer un objet temporaire contenant l'algorithme
  - Non transmissible d'un travail à un autre
  - A détruire après utilisation ou fermer le travail

Description	Utilisation	Type
Description du format de l'algorithme	Entrée	Char(*)
Nom du format	Entrée	Char(8)
Jeton du contexte de l'algorithme	sortie	Char(8)
Code erreur	E/S	Char(*)

Opérations sur le KeyStore : création des jetons

- La description dépend du format, qui dépend de l'algorithme
- Nom du format :

algorithme	Valeur
Chiffrement par blocs (DES, Triple DES, AES, et RC2)	ALGD0200
Chiffrement en flux (RC4-compatible)	ALGD0300
Algorithme à clé publique (RSA)	ALGD0400
Algorithme de hashage (MD5, SHA-1, SHA-256, SHA-384, SHA-512)	ALGD0500

Opérations sur le KeyStore : création des jetons

- Description du format ALGD0200 :

Description	Type	Valeur possible
Algorithme	Binaire(4)	20: DES, 21: 3DES, 22: AES, 23: RC2
Longueur du bloc	Binaire(4)	8: DES, 3DES, RC2, 16,24 ou 32: AES
Mode	Char(1)	0: ECB, 1:CBC, 2: OFB, 3:CFB (1 bit), …
Option de remplissage	Char(1)	0: pas de remplissage, 1 : caractère…
Caractère de remplissage	Char(1)	Un caractère
Réservé	Char(1)	X'00'
Longueur MAC	Binaire(4)	Longueur du code d'authentification du message
Taille réelle de la clé	Binaire(4)	1-1024: RC2, 0 : DES, 3DES, AES
Vecteur d'initialisation	Char(32)	

- Description du format ALGD0300 :

Description	Type
Algorithme	Binaire(4)

Opérations sur le KeyStore : création des jetons

- Exemple d'un format Triple DES :

```
D CrtAlgCtx       Pr                  ExtProc('Qc3CreateAlgorithm-
 D                                       Context')
 D  AlgDsc                      1024a   Const  Options( *VarSize )
 D  AlgDscFmt                      8a   Const
 D  AlgCtxTkn                      8a
 D  Error                      32767a          Options( *VarSize )
**-- Block cipher algorithm description:
D ALGD0200        Ds                    Qualified
D  BlkCphAlg                    10i 0
D  BlkLen                       10i 0
D  Mode                          1a
D  PadOpt                        1a
D  PadChr                        1a
D  Rsv                           1a
D  MacLen                       10i 0
D  EfcKeySiz                    10i 0
D  InzVct_IV                    32a
…
/free
ALGD0200.BlkCphAlg = 21;
   ALGD0200.BlkLen    = 8;
   ALGD0200.Mode      = '0';
   ALGD0200.PadOpt    = '1';
   ALGD0200.PadChr    = x'00';
   ALGD0200.Rsv       = x'00';
   ALGD0200.MacLen    = *Zero;
   ALGD0200.EfcKeySiz = *Zero;
If  %Parms >= 6;
     ALGD0200.InzVct_IV = PxInzVct;
   Else;
     ALGD0200.InzVct_IV = *Allx'00';
   EndIf;
/end-free
 CrtAlgCtx( ALGD0200 : 'ALGD0200' : AlgCtxTkn : ERRC0100 );
```

Opérations sur le KeyStore : création des jetons

- API Qc3CreateKeyContext ou QC3CRTKX
  - Programme de service **QC3CTX (*USE)**
  - Permet de créer un objet temporaire contenant l'algorithme

Description	Utilisation	Type
Clé	Entrée	Char(*)
Longueur de la chaine contenant la clé	Entrée	Binaire(4)
Format de la clé	Entrée	Char(1)
Type de la clé	Entrée	Binaire(4)
Forme de la clé (cryptée ou en clair)	Entrée	Char(1)
Clé d'encryptage	Entrée	Char(*)
Algorithme de cryptage	Entrée	Char(8)
Jeton de contexte de la clé	Sortie	Char(8)
Code erreur	E/S	Char(*)

Opérations sur le KeyStore : création des jetons

- Longueur de la chaîne contenant la clé
  - Pas forcément identique à la longueur de la clé
    - Clé binaire (format 0) et en clair : même longueur
    - Clé binaire (format 0) et cryptée : la longueur de la clé sera la longueur de la clé en clair (décryptée)
    - Clé stockée dans un keystore (format 4, label) : la longueur de la clé sera de 56
      - Nom qualifié du keystore – char(20)
      - Label – char(32)
      - Zone réservée – char(4) : nulle x'00'
- Pour les autres paramètres, se référer à la documentation ou au jeton de contexte de l'algorithme

Opérations sur le KeyStore : création des jetons

- Exemple d'une clé pour un algorithme Triple DES :

```
P GetKeyCtx       B                   Export
 D                 Pi             8a
 D  PxKeyStr                     64a   Varying  Const
 D  PxKeyFmt                      1a   Const
 D  PxKekKeyCtxTk                 8a   Const  Options( *NoPass )
 D  PxKekAlgCtxTk                 8a   Const  Options( *NoPass )

 **-- Local declarations
 D KeyCtxTkn       s              8a
 D Lg              s              8  0
/Free
Lg = %Len( PxKeyStr );

    If  %Parms >= 3;

      CrtKeyCtx( PxKeyStr : %Len(PxKeyStr): PxKeyFmt : 21: '1' : PxKekKeyCtxTk
              : PxKekAlgCtxTk  : KeyCtxTkn  : ERRC0100 );
   Else;

     CrtKeyCtx( PxKeyStr : %Len(PxKeyStr) : PxKeyFmt : 21 : '0' : ' '
              : ' ' : KeyCtxTkn  : ERRC0100);
   EndIf;

   If  ERRC0100.BytAvl > *Zero;
     Return  *Blanks;
    Else;
      Return  KeyCtxTkn;
    EndIf;
  /End-Free
P GetKeyCtx       E
```

Opérations sur le KeyStore : destruction des jetons

- API Qc3DestroyAlgorithmContext ou QC3DESAX
  - Programme de service **QC3CTX (*USE)**

Description	Utilisation	Type
Jeton de contexte d'algorithme	Entrée	Char(8)
Code erreur	E/S	Char(*)

- API Qc3DestroyKeyContext ou QC3KEYAX
  - Programme de service **QC3CTX (*USE)**

Description	Utilisation	Type
Jeton de contexte de clé	Entrée	Char(8)
Code erreur	E/S	Char(*)

Tous le reste n'est que du cryptage de données

- API Qc3EncryptData ou QC3ENCDT
  - Programme de service **QC3DTAEN(*USE)**

Description	Utilisation	Type
Donnée en clair	Entrée	Char(*)
Longueur de la donnée en clair	Entrée	Binaire(4)
Format de la donnée	Entrée	Char(8)
Description de l'algorithme	Entrée	char(*)
Nom de la description	Entrée	Char(8)
Description de la clé	Entrée	Char(*)
Format de la description de la clé	Entrée	Char(8)
Fournisseur de service cryptographique	Entrée	Char(1)
Unité de cryptographie	Entrée	Char(10)
Donnée cryptée	Sortie	Char(*)
Longueur de la zone de réception de la donnée cryptée	Entrée	Binaire(4)
Longueur de la donnée cryptée retournée	Sortie	Binaire(4)
Code erreur	E/S	Char(*)

Prototype

- Très long…

```
D EncryptData     Pr                  ExtProc( 'Qc3EncryptData' )
D  ClrDta                   65535a    Const  Options( *VarSize )
D  ClrDtaLen                   10i 0 Const
D  ClrDtaFmt                    8a   Const
D  AlgDsc                    1024a   Const  Options( *VarSize )
D  AlgDscFmt                    8a   Const
D  KeyDsc                    1024a   Const  Options( *VarSize )
D  KeyDscFmt                    8a   Const
D  CrpSrvPrv                    1a   Const
D  CrpDevNam                   10a   Const
D  EncDta                   65535a         Options( *VarSize )
D  EncDtaLen                   10i 0 Const
D  EncRtnLen                   10i 0
D  Error                    32767a         Options( *VarSize )
```

Prototype

- Mais peut être simplifié, comme on utilise souvent le même algorithme

```
 D EncDtaStr       Pr           1024a   Varying
 D  PxDtaStr                    1024a   Varying  Const
 D  PxAlgCtxTkn                    8a
 D  PxKeyCtxTkn                    8a
…
 P EncDtaStr       B                    Export
 D                 Pi           1024a   Varying
 D  PxDtaStr                    1024a   Varying  Const
 D  PxAlgCtxTkn                    8a
 D  PxKeyCtxTkn                    8a

 **-- Algorithm context token structure:
 D ALGD0100        Ds                   Qualified
 D  AlgCtxTkn                      8a
 D  FinOprFlg                      1a
 **-- Local declarations
 D EncDtaStr       s            1024a
 D EncRtnLen       s              10i 0
```

Prototype

- Suite…

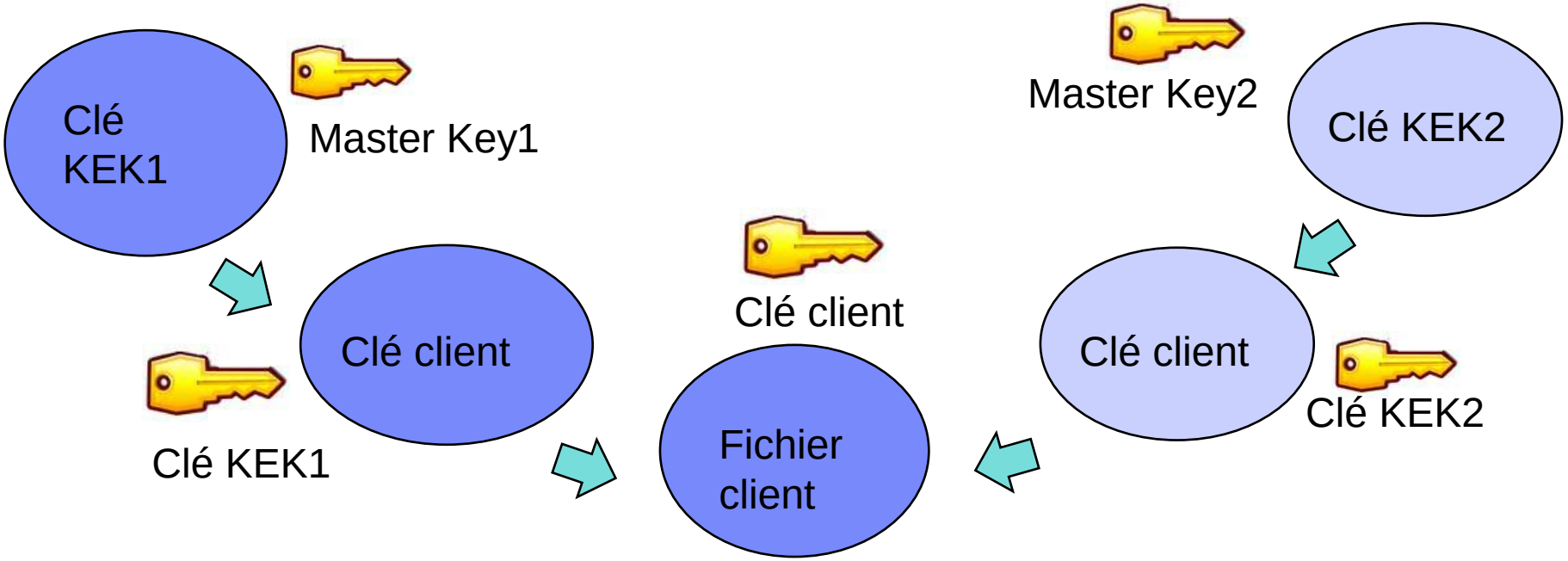
```
/Free

    ALGD0100.AlgCtxTkn = PxAlgCtxTkn;
    ALGD0100.FinOprFlg = '1';

    EncryptData( PxDtaStr: %Len( PxDtaStr ): 'DATA0100': ALGD0100
               : 'ALGD0100': PxKeyCtxTkn : 'KEYD0100' : CSP_SFW
               : *Blanks : EncDtaStr  : %Size( EncDtaStr )
               : EncRtnLen
               : ERRC0100
               );

   If  ERRC0100.BytAvl > *Zero;
     Return  NULL;

   Else;
     Return  %Subst( EncDtaStr: 1: EncRtnLen );
   EndIf;

 /End-Free

P EncDtaStr       E
```

S22 - Cryptographie

PROCEDURES DE SAUVEGARDE

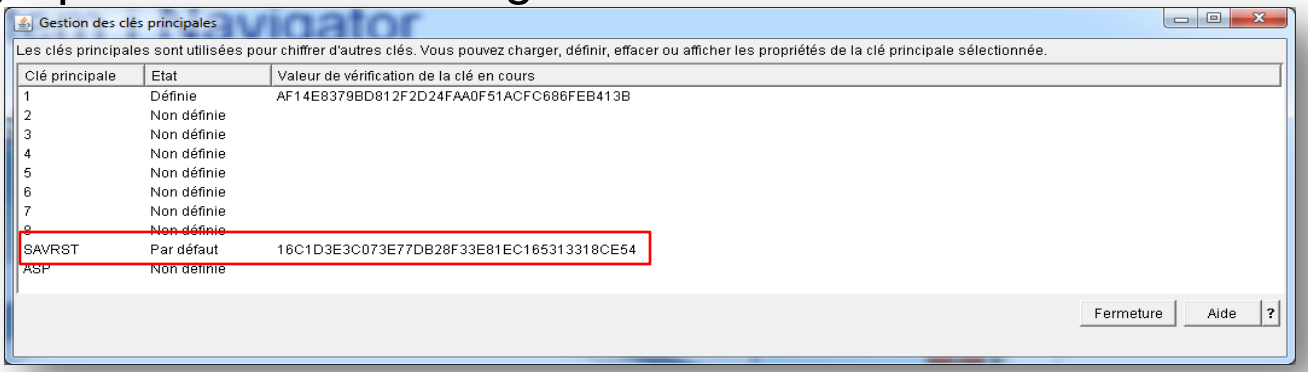
Sauvegardes

- Les fichiers (KeyStores et fichiers clients cryptés)
  - Sauvegarde classique
  - Attention ! Les Keystores dépendent des Master Keys pour être utilisables
  - Les fichiers clients cryptés par les clés client n'ont pas cette contrainte



Sauvegardes

- Les Masters Keys
  - Sauvegarde par un SAVSYS uniquement, avec cryptage des Master Keys par la clé de sauvegarde/restauration



  - La restauration ne sera possible que si la clé SAVRST est la même entre la sauvegarde et le système.
    - Si pas d'égalité, les clés restaurées sont en instance.
    - Si égalité, la clé restaurée deviendra la version CURRENT, et la version CURRENT deviendra la version OLD.
    - Les éléments cryptés par la Master Key devront probablement être traduits (APIs Qc3Translate…)

S22 - Cryptographie

ET SI JE CHANGE DE MACHINE?

Pour passer d'une machine à une autre

- Les masters Keys ne peuvent pas être copiées !!
  - Les KeyStores dépendant des clés maîtresses… Problème !
- Solution 1 : ne pas utiliser les Master Keys et les KeyStores
  - N'y songez même pas, Sauf si la première machine n'en a pas
- Solution 2 : saisir la même Master Key sur l'autre machine
  - Il faut donc une nouvelle saisie avec le même programme "maison"
- Solution 3 : créer une master key supplémentaire le temps de la migration
  - Créer une nouvelle master Key (temporaire) identique sur les 2 machines
  - Dupliquer le keystore par un CRTDUPOBJ et le traduire vers la nouvelle Master Key (Qc3TranslateKeyStore)
  - Le déplacer vers le nouveau système et traduire le KeyStore vers une master Key définitive

Les APIs à connaître

API ILE	API OPM	Description
Qc3EncryptData	QC3ENCDT	Cryptage des données
Qc3DecryptData	QC3DECDT	Décryptage des données
Qc3TranslateData	QC3TRNDT	Traduction d'éléments cryptés par une clé vers une autre clé
Qc3CreateKeyContext	QC3CRTKX	Création d'un contexte de clé. Permet de virtualiser la clé
Qc3CreateAlgorithmContext	QC3CRTAX	Création d'un contexte d'algorithme.
Qc3CreateKeyStore	QC3CRTKS	Création d'un keystore (magasin de clés)
Qc3WriteKeyRecord	QC3WRTKR	Enregistrement d'une clé dans un magasin de clés
Qc3DeleteKeyRecord	QC3DLTKR	Suppression d'une clé d'un magasin de clés
Qc3ClearMasterKey	QC3CLRMK	Mise à blanc d'une Master Key
Qc3ExportKey	QC3EXPKY	Décryptage d'une clé cryptée par une Master Key et encryptage par une autre clé (pas une master Key)
Qc3ImportKey	QC3IMPKY	Cryptage d'une clé par une master Key
Qc3LoadMasterKeyPart	QC3LDMKP	Chargement des composants d'une Master Key (NEW)
Qc3SetMasterKey	QC3SETMK	Affectation des composants chargés à la nouvelle Master Key
Qc3TranslateKeyStore	QC3TRNKS	Traduit un KeyStore d'une Master Key vers une autre ou vers la nouvelle version de celle-ci (OLD vers CURRENT)

Conseils et bonnes pratiques

- **Utiliser des clés en plusieurs parties**
  - **Décomposer votre clé en 2, 3, … n parties**
  - **Chaque partie est connue par une personne différente**
  - **Chaque partie peut être générée aléatoirement**
    - **On évite ainsi les dates de naissance des enfants, le prénom du chien, …**
  - **Chaque partie peut subir un traitement supplémentaire**
    - **Souvent un XOR avec les autres parties**
  - **Chaque partie est stockée séparément des autres**
    - **Dans des dossiers, des fichiers, des magasins de clés, …**
- **Changez régulièrement vos clés !!**
- **Une Master Key ne se prête pas**
  - **Elle est propre à une machine**
  - **Utilisez des Master Keys temporaires pour les migrations**

Conseils et bonnes pratiques

- **Adoptez un haut niveau de cryptage (algorithme)**
- **Rajoutez votre touche personnelle**
  - **Traitement supplémentaires, cryptage multi-niveaux, …**
- **A chaque ajout d'une clé dans un KeyStore**
  - **Sauvegardez votre KeyStore**
- **Ne jamais traduire un KeyStore**
  - **Faire une duplication et traduire le duplicat**
- **Ne jamais sauvegarder une version cryptée des passphrases**
  - **Les stocker à l'extérieur de la machine dans un coffre**

Les questions à se poser

- **Où sont stockées les clefs ?**
  - **Fichiers, magasins de clés, …**
- **Comment sont générées les clés ?**
  - **Programmes maison, outils de prestataires, …**
- **Comment sont-elles protégées ?**
- **Qui les gère ?**
- **Rythme de modification ?**
- **Comment sont-elles utilisées dans les applications ?**
  - **Transit en clair, boîte noire, …**
- **Stratégie de sauvegarde/restauration ?**
  - **Clés et données cryptées sur des supports distincts**

Informations utiles

- Ressources
  - IBM Systems – iSeries Cryptographic Services APIs
  - IBM iSeries Wired Connection 5.1- DCM & crypto services sg246168
    - http://www.redbooks.ibm.com/abstracts/sg246168.html
  - Protecting i5OS Data with Encryption sg247399
    - http://www.redbooks.ibm.com/abstracts/sg247399.html
  - Security Guide for IBM I V6 1 sg247680
  - http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0
- Forums
  - http://forum.xdocs400.com
  - http://www.developpez.net
  - http://forum.commonfr.org
  - http://www.volubis.fr
  - http://www.know400.fr

Avons-nous vraiment le temps pour des questions ?

- Le temps manque(ra) donc :
  - Habib SAAD jmsaad@gaia.fr

- Sinon, vous avez des questions ?