IBM Systems Lab Services & Training

Introduction of encryption functions and services in IBM i (i5/OS)

EUROPE LUXEMBOURG

Thomas Barlen
Consulting IT Specialist
IBM Systems Lab Services and Training

Helping our clients **WIN the race**

Notices

Trademarks

Acknowledgements

- This presentation was developed by Thomas Barlen, IBM Systems Lab Services and Training. Thomas is based in Germany, but works world-wide on mostly IBM i (i5/OS) related security projects and presents at technical conferences.

- You can also engage Thomas for any kind of IBM i (i5/OS) related security (including but not limited to base security, object level access, object signing, IFS security, cryptography, security assessments, etc.) project or issue as well as cross-platform single signon projects.

- The best way to reach Thomas is by e-mail at barlen@de.ibm.com.

IBM Systems Lab Services and Training

IBM Systems Lab Services and Training can help you optimize the utilization of your data center and system solutions.

Lab Services has the knowledge and deep skills to support you through the entire information technology race. Focused on the delivery of new technologies and niche offerings, Lab Services and Training collaborates with IBM Global Services and IBM Business Partners to provide complementary services that will help lead through the turns and curves to keep your business running at top speed.

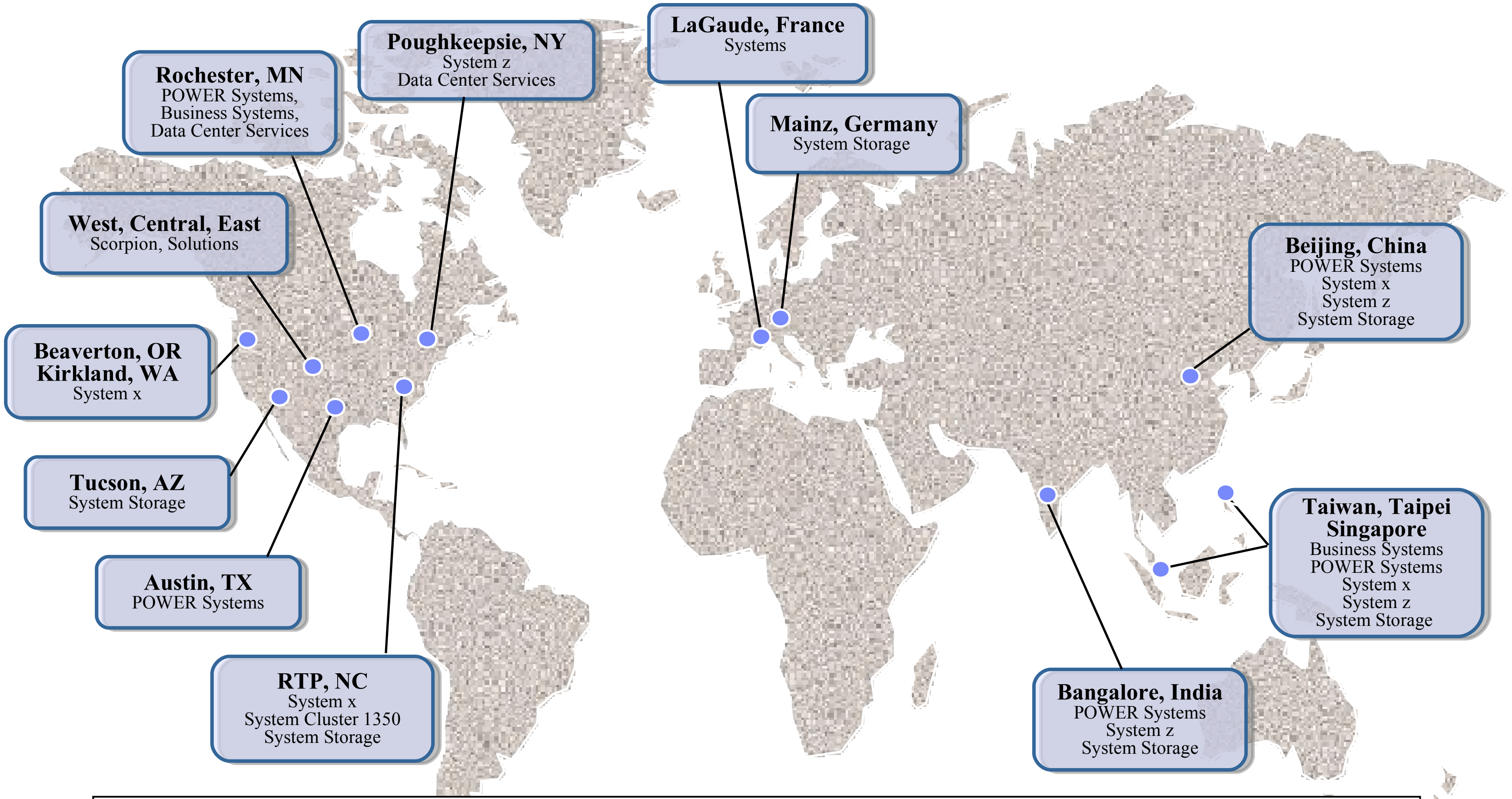
The team consists of consultants and specialists from various geographies with extensive field experience and roots into the development labs.

What we offer

We apply the intellectual and technical capital of IBM development toward making sure IBM Systems products succeed, build loyalty and increase satisfaction. Our services include:

- Implementation, development and solution integration for IBM Power Systems, IBM System i™, IBM System p™, IBM System x™, and IBM System z™ technology.Storage services to simplify data migration to IBM System Storage™, enable and automate advanced hardware functions, implement virtualization technology for improving SAN-based storage and data management, and deploy copy services and disaster recovery solutions in an open systems environment.
- Application development, such as software components to accelerate solution deployment.
- IT optimization studies to help simplify, optimize, and reduce costs.
- Cross-platform virtualization, automation and systems management solutions.
- Data Center evaluation, consultation, and direction in power, packaging and cooling.
- Clusters support to serve a broad range of functions, from server/workload consolidation to high-performance parallel computing tasks.

Worldwide STG Lab Services Delivery Teams: a global team



Our 400 consultants reside in these centers as well as other cities around the world

STG Lab Services Key Offerings

✓ New Offering in 2008

Power Systems

- Copy Services
- HA / Business Continuity, Performance
- **System and Application Security**
- **Security Assessments**
- **Security Education**
- **Single Sign-on (SSO)**
- **Cryptography**
- **Compliance**
- SOA
- AIX
- ✓ Power Care for High End POWER Systems
- ✓ Storage Migration for i5/OS clients

Modular Systems

- System x Server & Storage Implementation & Skills Transfer
- System Management
- Virtualization, SCON, Migration
- High Availability
- Project Management (System Cluster 1350)
- ✓ iDataplex
- ✓ Blades Open Fabric Manager
- ✓ Virtualization Alternatives to VMware
- ✓ BCU Services

IT Optimization / Virtualization

- IT Systems Rationalization Study
- IT Systems Energy Rationalization Study
- ILM Assessments and Workshops
- Advanced Virtualization Rapid Assessment
- COBRA Rapid Assessment
- IT Systems Energy Efficiency Rapid Assessment
- Virtualization Rapid Assessment for UNIX
- Rapid Assessment for Linux on System z
- ✓ IT Current State Assessment
- ✓ Application IT Infrastructure Study
- ✓ Blue Cloud Offerings

Mainframe

- Linux on System z
- Java Performance Optimization
- System z SOA Services
- System z Currency and Migration Jumpstarts
- System z Performance Assessment for Parallel Sysplex middleware
- ✓ PCI Compliance Mitigation Services
- ✓ RACF Health Check
- ✓ DFSMS/DFSMShsm Health Assessment
- ✓ STP (Sysplex Timer Protocol) Planning and implementation

System Storage

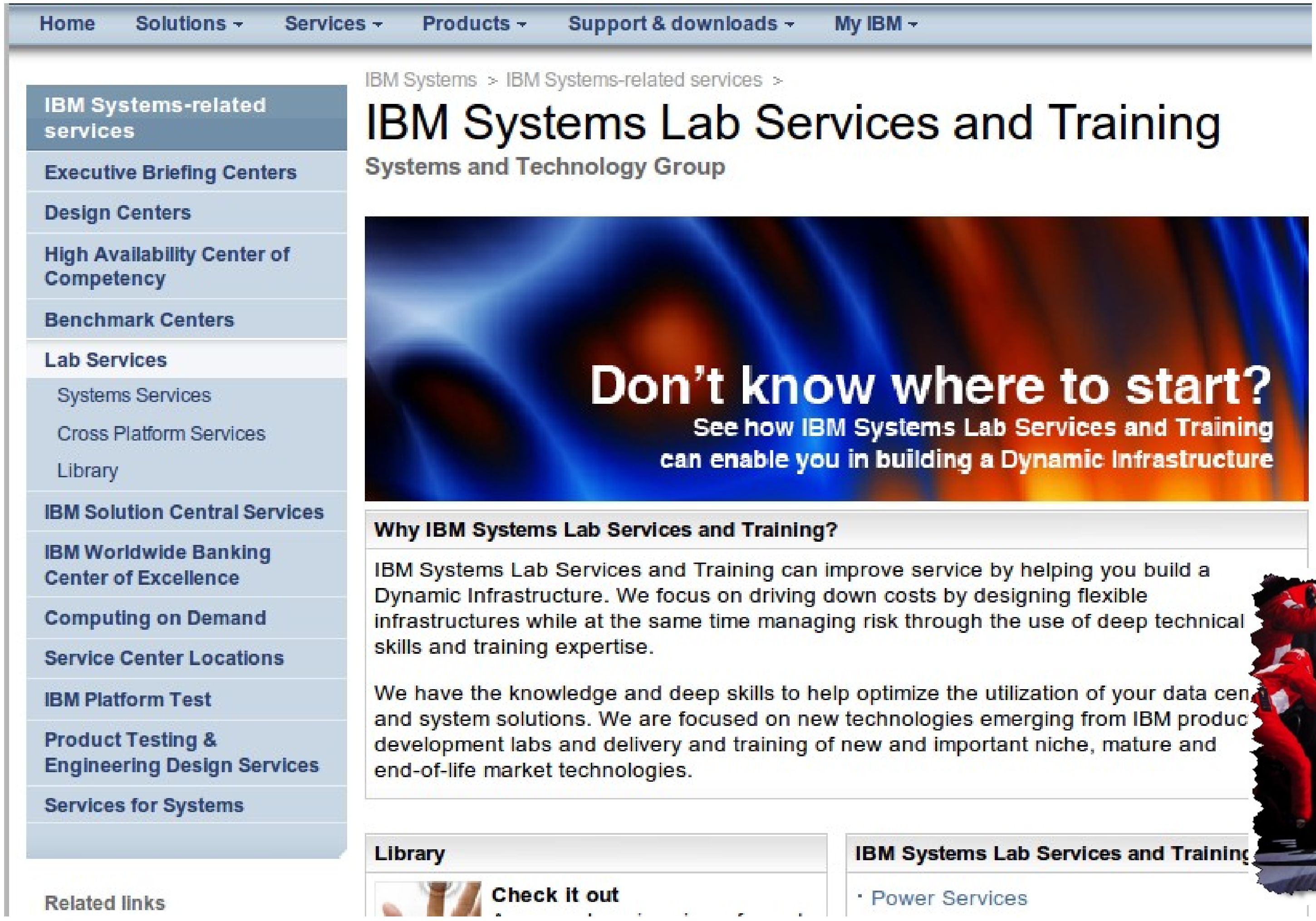
- Technical Project Management
- Storage Consulting
- SAN Services
- SVC Services
- Advanced Copy Services
- ✓ Softek
- ✓ XIV
- ✓ N Series
- ✓ Novus
- ✓ Secure Data Erase
- ✓ Tape Encryption Key Management (TEKM)

Data Center Services

- Trends & Best Practices
- IT Systems Energy Efficiency Assessment
- Scorpion w/ Power
- Thermal Analysis
- Power Analysis
- Assessment of Cool Blue Rear Door Heat Exchanger (RDHX)
- ✓ MMT Thermal Offering
- ✓ IBM System Director Advanced Energy Manager Implementation Jumpstart
- ✓ EITA
- ✓ Data Center Planning Offerings

How to involve IBM Systems Lab Services

- Visit out Web site at http://www.ibm.com/systems/services/labservices/



Home | Solutions | Services | Products | Support & downloads | My IBM

IBM Systems-related services
- Executive Briefing Centers
- Design Centers
- High Availability Center of Competency
- Benchmark Centers
- Lab Services
  - Systems Services
  - Cross Platform Services
  - Library
- IBM Solution Central Services
- IBM Worldwide Banking Center of Excellence
- Computing on Demand
- Service Center Locations
- IBM Platform Test
- Product Testing & Engineering Design Services
- Services for Systems

Related links

IBM Systems > IBM Systems-related services >

IBM Systems Lab Services and Training

Systems and Technology Group

Don’t know where to start?
See how IBM Systems Lab Services and Training can enable you in building a Dynamic Infrastructure

Why IBM Systems Lab Services and Training?

IBM Systems Lab Services and Training can improve service by helping you build a Dynamic Infrastructure. We focus on driving down costs by designing flexible infrastructures while at the same time managing risk through the use of deep technical skills and training expertise.

We have the knowledge and deep skills to help optimize the utilization of your data cen and system solutions. We are focused on new technologies emerging from IBM produc development labs and delivery and training of new and important niche, mature and end-of-life market technologies.

Library

Check it out

IBM Systems Lab Services and Training

- Power Services

Agenda

- Introduction to Cryptography
  - Terminology
  - Major drivers for forcing companies to implement encryption
  - Key management
- Network encryption
  - Available services and functions
  - Implementation prerequisites and tasks
- Backup encryption
  - Overview of software and hardware-based backup encryption functions
- ASP Encryption
- Database encryption
  - Overview of available encryption services
  - Implementation prerequisites and planning

Introduction

Defining Cryptography

A process associated with scrambling plaintext (or clear text) into cipher text (a process called encryption), then back again into plaintext (known as decryption)

Notes:

Cryptography

- Cryptography is the science of keeping data secure. Cryptography allows you to store information, or to communicate with other parties, while preventing non-involved parties from understanding the stored information or understanding the communication. Encryption transforms understandable text into an unintelligible piece of data (ciphertext). Decryption restores the understandable text from the unintelligible data. Both processes involve a mathematical formula, or algorithm, and a secret sequence of data (the key).

The Need for Encryption – Major Drivers

- Why do you need to care for encryption?
  - to protect sensitive information
    - from industrial espionage
    - from unauthorized use when stolen
    - while stored on disk or backup media or traversing the network
  - because it is required (implicitly or explicitly)
    - by laws and government regulations
    - Payment Card Industry (PCI) Data Security Standard
    - HIPPA (Health Insurance Portability and Accountability Act)
    - Sarbanes Oxley Act
    - and many more…..

Encryption is the ultimate protection mechanism because even if someone breaks through all other protection mechanisms and gains access to encrypted data, they will not be able to read the data without further breaking the encryption.
Source: PCI Data Security Standard, Protect Stored Data

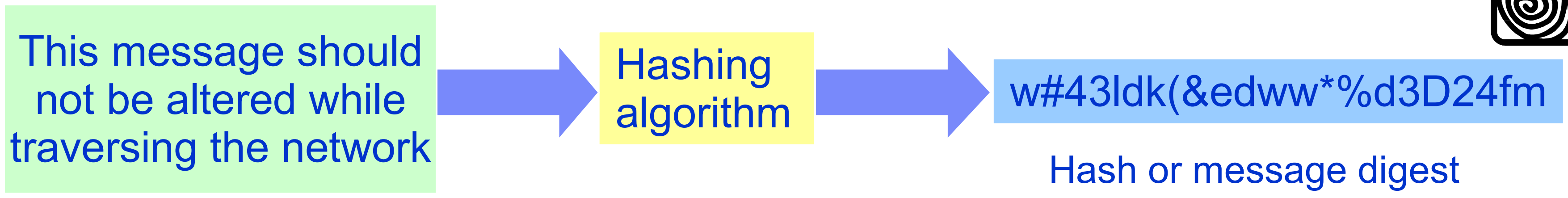
Notes:

- Nowadays, most companies still do not encrypt data on disk or while stored on backup media. However, several legislative or industry-specific requirements dictate implicitly or explicitly the encryption of data. For example, the Visa Payment Card Industry (PCI) states in paragraph 3.4:
- 3.4 Render sensitive cardholder data unreadable anywhere it is stored (including data on portable media, backup media, in logs, and data received from or stored by wireless networks) by using any of the following approaches:
  - One-way hashes (hashed indexes), such as SHA-1
  - Truncation
  - Index tokens and PADs, with the PADs being securely stored
  - Strong cryptography, such as Triple-DES 128-bit or AES 256-bit with associated key management processes and procedures.
  - The MINIMUM account information that needs to be rendered unreadable is the payment card account number.
- 3.5 Protect encryption keys against both disclosure and misuse.
  - 3.5.1 Restrict access to keys to the fewest number of custodians necessary
  - 3.5.2 Store keys securely in the fewest possible locations and forms.
- 3.6 Fully document and implement all key management processes and procedures, including:
  - 3.6.1 Generation of strong keys
  - 3.6.2 Secure key distribution
  - 3.6.3 Secure key storage
  - 3.6.4 Periodic key changes
  - 3.6.5 Destruction of old keys
  - 3.6.6 Split knowledge and dual control of keys (so that it requires 2 or 3 people, each knowing only their part of the key, to reconstruct the whole key).
  - 3.6.7 Prevention of unauthorized substitution of keys
  - 3.6.8 Replacement of known or suspected compromised keys

  Source: Visa PCI Standard
- HIPPA (Health Insurance Portability and Accountability Act) is another example where encryption of data should be used.

Terminologies and Technologies involved with Encryption - Hashing

- The mathematical process that produces a message digest or *hash*. Due to the mathematical algorithms involved, the original message cannot be recreated from the hash.
- Used to create a result character string of a given information to ensure integrity
  - for object and data integrity
  - e-mail integrity (signing an e-mail)
    - With signatures, hashing and encryption is involved
- Various hashing algorithms are available
  - e.g. SHA (Secure Hash Algorithm), MD5 (Message Digest 5)
  - SHA-1 is widely used in protocols, such as SSL, TLS, IPSec protocol framework
    - user's are advised to move towards SHA-256, SHA-384, or SHA-512


This message should not be altered while traversing the network

Hashing algorithm

w#43ldk(&edww*%d3D24fm

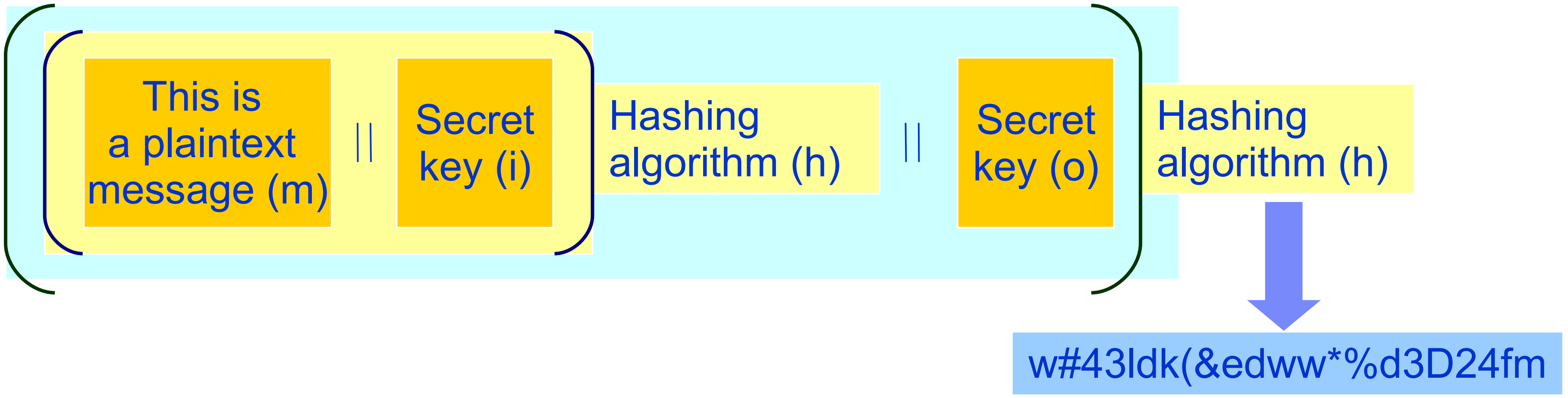
Hash or message digest

Notes:

- Hashing algorithms provide a one-way hash function to produce a fixed-length output string from a variable-length input string. For all practical purposes, one-way hashes are irreversible. This property makes them useful for authentication purposes. They are often used when generating digital signatures or other integrity checking processes.
- If users have a choice, they should move towards the use of SHA-256, SHA-384, or SHA-512, because MD5 as well as SHA-1 (in the year 2005) were successfully attacked. However, there is no reason to panic, experts just recommend moving towards a more stronger algorithm.

Terminologies and Technologies involved with Encryption - HMAC

- A keyed-hash message authentication code (HMAC), is a type of message authentication code (MAC) calculated using a cryptographic hash function in combination with a secret key.
- Cryptographic hash algorithms, such as SHA or MD5, can be used in the calculation of an HMAC



|| = concatenation (i) = inner key XORed 0x36 (o) = outer key XORed 0x5C

Notes:

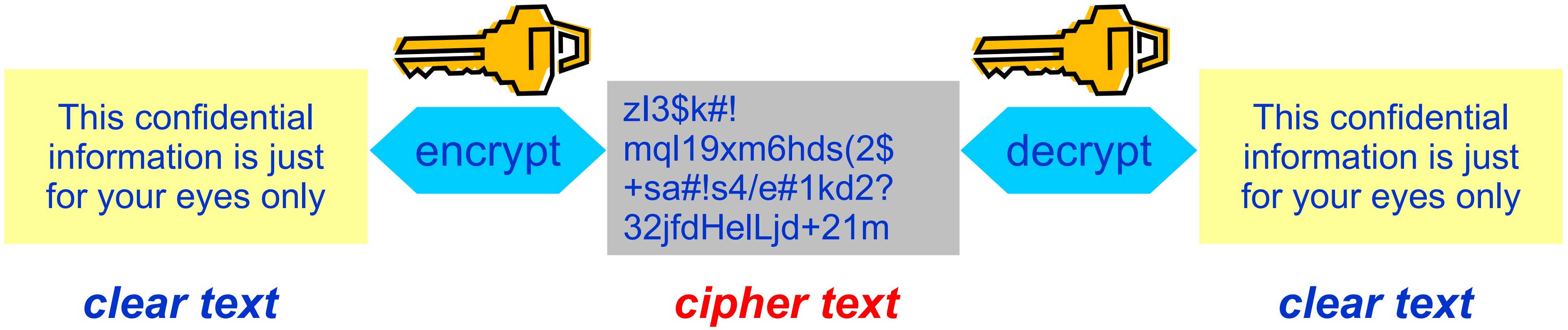
- A Hashed Message Authentication Codes (HMAC) is a type of message authentication code that is calculated by using a cryptographic hash function in combination with a secret key.It can be used to verify both the data integrity and the authenticity of a message. hash algorithms, such as MD5 or SHA-1, can be used in the calculation of an HMAC. The result of an HMAC operation is called HMAC-MD5 or HMAC-SHA-1 accordingly. The cryptographic strength of the HMAC depends on the cryptographic strength of the underlying hash function and on the size and quality of the key.

- An iterative hash function breaks up a message into blocks of a fixed size and iterates over them with a compression function. For example, MD5 and SHA-1 operate on 512-bit blocks. The size of the output of HMAC is the same as that of the underlying hash function (128 or 160 bits in the case of MD5 and SHA-1), although it can be truncated if desired.

- HMAC calculation is performed as illustrated on the previous page where *h* is an iterated hash function, The secret key is padded with extra zeros to the block size of the hash function and *m* is the message to be authenticated. "||" denotes concatenation. The two secret keys (i) and (o) are actually built by performing an exclusive or operation with the secret key and a constant. The constants i and o, each one block long, are defined as i = 0x363636...3636 and o = 0x5c5c5c...5c5c. That is, if block size of the hash function is 512, *i* and *o* are 64 repetitions of the (hexadecimal) bytes 0x36 and 0x5c respectively.

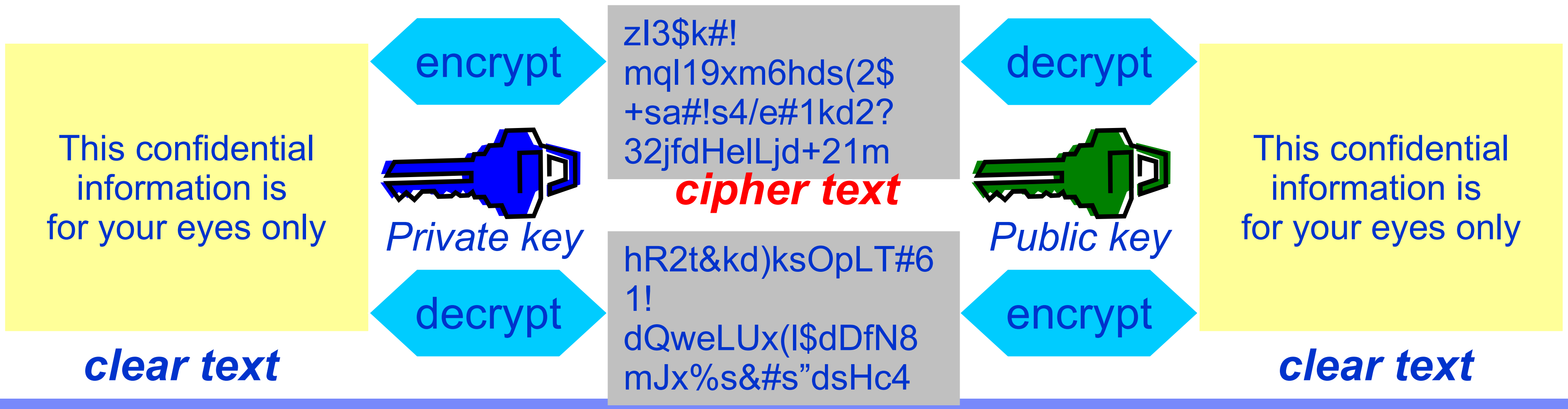
Terminologies and Technologies involved with Encryption – Symmetric Encryption

- With symmetric encryption, the same key (shared secret) is used to encrypt and decrypt data
- Commonly used symmetric algorithms include:
  - DES, 3DES, AES, RC2, RC4
- Faster than asymmetric encryption
- Typically used for data encryption


This confidential information is just for your eyes only
encrypt
zI3$k#! mqI19xm6hds(2$ +sa#!s4/e#1kd2? 32jfdHelLjd+21m
decrypt
This confidential information is just for your eyes only
clear text
cipher text
clear text

Terminologies and Technologies involved with Encryption – Asymmetric Encyrption

- With asymmetric (public key) encryption, a key pair (public/private key pair) is used to encrypt and decrypt data
  - Data encrypted with one key can only be decrypted with the corresponding second key
  - The two keys are mathematically related
- Commonly used asymmetric algorithm is RSA
- More compute intensive than symmetric encryption


This confidential information is for your eyes only
clear text
encrypt
Private key
decrypt
zI3$k#! mqI19xm6hds(2$ +sa#!s4/e#1kd2? 32jfdHelLjd+21m
cipher text
hR2t&kd)ksOpLT#6 1! dQweLUx(I$dDfN8 mJx%s&#s”dsHc4
decrypt
Public key
encrypt
This confidential information is for your eyes only
clear text

Notes:

- There are two types of cryptography:
  - In shared/secret key (symmetric) cryptography, one key is a shared secret between two communicating parties. Encryption and decryption both use the same key. In public key (asymmetric) cryptography, encryption and decryption each use different keys. A party has two keys: a public key and a private key. The two keys are mathematically related, but it is virtually impossible to derive the private key from the public key. A message that is encrypted with someone's public key can be decrypted only with the associated private key. Alternately, a server or user can use a private key to "sign" a document and use a public key to decrypt this digital signature. This verifies the document's source.
- Private key
  - A private key is one of an asymmetric key pair. A user or server can use a private key to decrypt messages that were encrypted with the corresponding public key. A user or server can also use a private key to encrypt messages that only the corresponding public key can decrypt. A public key is usually bound to the owner's digital certificate, and it is available for anyone to use. A private key, however, is protected by, and available only to, the owner of the key. This limited access ensures that communications that use the key are kept secure.
- Public key
  - A public key is one of an asymmetric key pair and is usually bound to the owner's digital certificate. Consequently, a public key is available for anyone to use. A public key consists of a data string and an algorithmic pattern.

Terminologies and Technologies involved with Encryption – Encryption versus Hashing

- Encryption and Hashing is mixed up very often
- Encryption
  - Information is made unreadable via the use of an algorithm and a key
  - Making the information unreadable is called encryption and corresponds to the goal of confidentiality
  - The same encryption algorithm and a key is used to decrypt the information or make it readable
- Hashing
  - Hashing is a kind of one-way encryption
  - Hashing algorithms generate a message digest or hash of a given information
  - The hash is not reversible into the original message
  - Primarily used to ensure

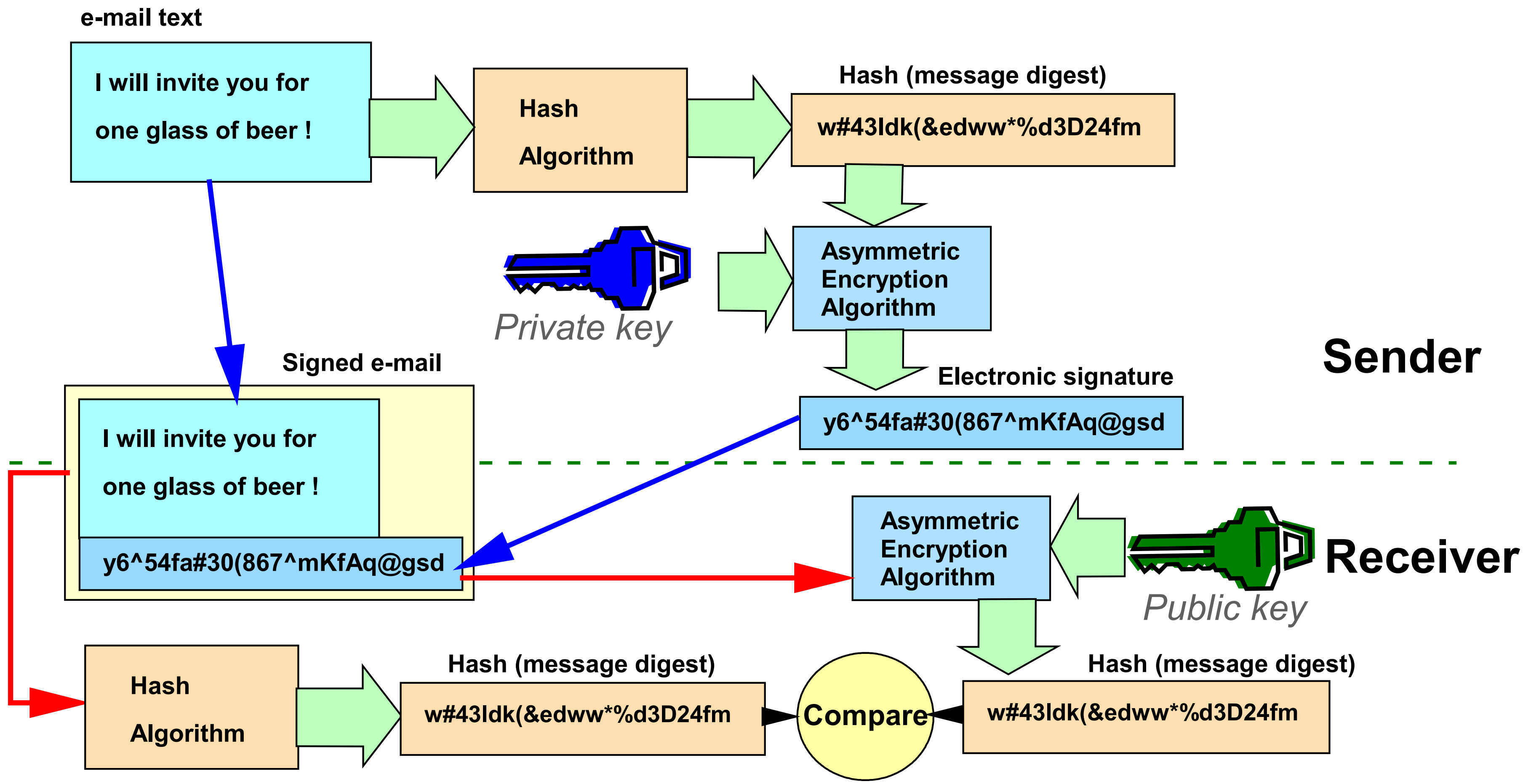
Confidentiality

Integrity

Notes:

- In many cases people mix up encryption and hashing. Even if both of these techniques are used to ensure a certain security goal, the purpose of each of them is different.
- Encryption is primarily used to scramble data or make them unreadable using an encryption algorithm in combination with an encryption key. The encrypted message can be decrypted using the same encryption algorithm and the same key or a corresponding key of a key pair, depending on the encryption method. Hashing is used to produce a fixed-length digest of a given message. The hash or digest cannot be reverse-engineered to obtain the original message.

Example: Electronic Signatures with RSA



Notes:

- You can protect data from undetected changes by including a proof of identity value called a digital signature. A digital signature relies on hashing and public key cryptography. When you sign data, you hash the data and encrypt the results with your private key. In case of RSA, the encrypted hash value is called a digital signature.
- If you change the original data, a different digital signature will be generated.
- Verifying a digital signature is the opposite of signing data. Verifying a signature will tell you if the signed data has changed or not. When a digital signature is verified, the signature is decrypted using the public key to produce the original hash value. The data that was signed is hashed. If the two hash values match, then the signature has been verified.

Introduction to Key Management

Key Management

Encryption requires the use of encryption keys

- Why is key management important?

If encryption keys are hardcoded in applications, thus automatically performing encryption/decryption or can easily be accessed by users or applications, encryption of information in the database does not provide a higher level of security at all!

Key management issues

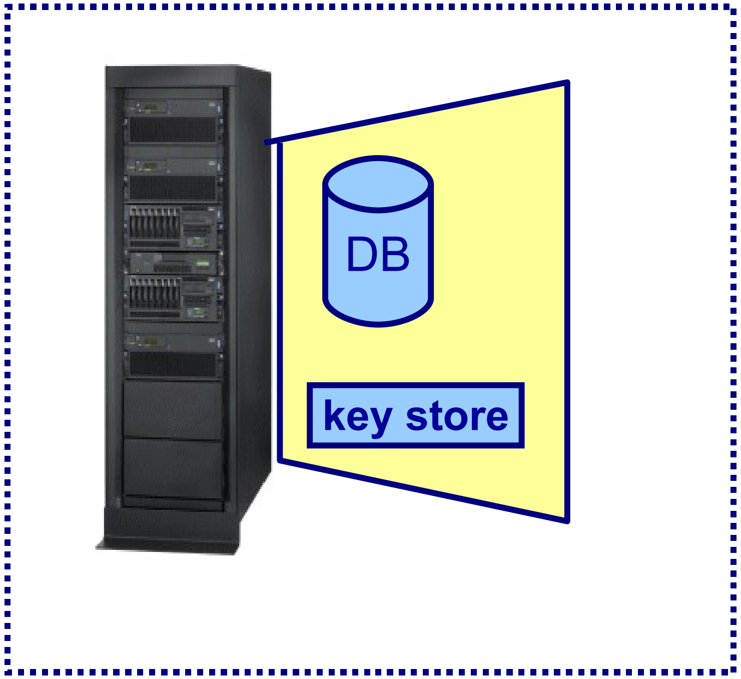
- Where is the encryption key stored
- How are encryption keys generated
- How is the encryption key protected
- How is the encryption key retrieved and used by applications
- How often will the encryption key change
- Who manages encryption keys
- What is the backup strategy for encryption keys

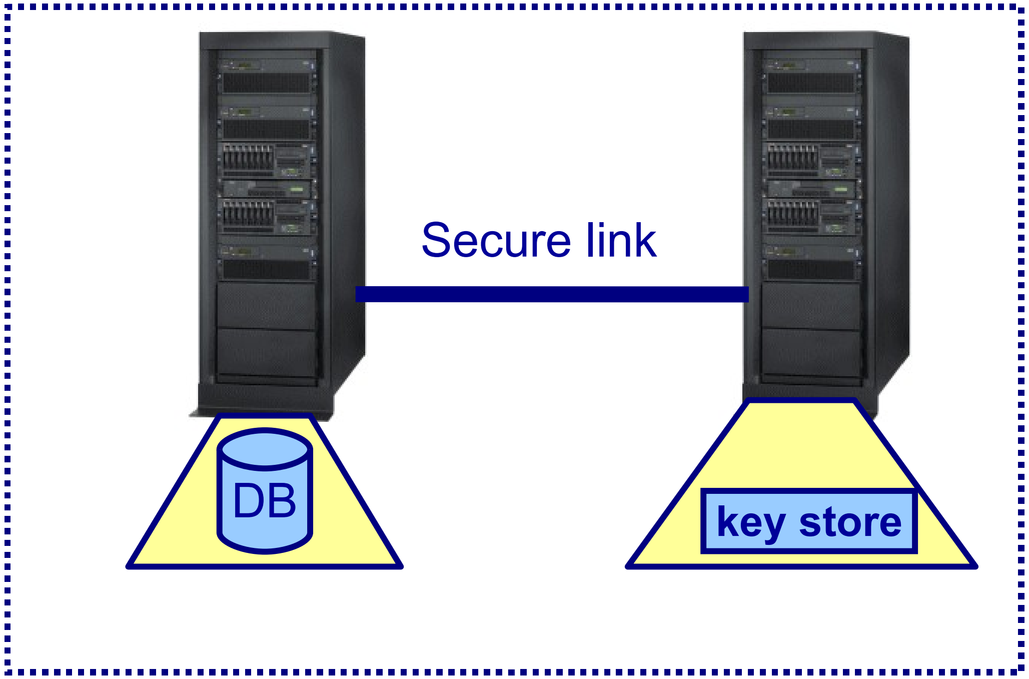
Notes:

- One basic rule with cryptography is that encrypted data is only as secure as the protection of the encryption keys. For example, if you store encryption keys in clear-text on disk or together with an encrypted backup on the backup media, encryption does not buy you any protection, because a potential attacker can easily decrypt the data.
- There are several issues associated with key protection. For example, you need to plan for the key store location, how encryption keys are generated, how often encryption keys are changed, or who is responsible for key management.

Key Store Location

- Several options conceivable, some are purely software-based, others leverage special hardware security modules (HSMs)
- Starting from i5/OS V5R4, integrated key management APIs and objects are available for simplifying key management
  - applies to software-based encryption
  - the 4764 cryptographic coprocessor manages keys within the adapters hardware domain (more later...)



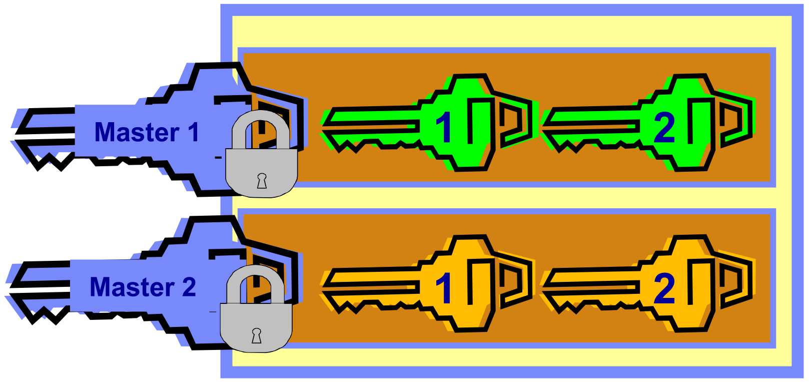


Notes:

- It is of utmost importance to protect encryption keys. When stored on the system that also contains the encrypted data, the encryption keys should be stored encrypted. The key to decrypt the encryption keys should not be stored on that system. This key could be entered manually or retrieved through a secure channel from another device.

IBM i Cryptographic Services Key Management

- Starting from V5R4 software-based encryption can utilize the i5/OS Cryptographic Services Key Management (CKM) functions for managing and protecting keys
- Key store files are encrypted with master keys
  - master keys are maintained and stored in a protected area in the Licensed Internal Code (LIC)
- V5R4: Key management can only be done through APIs
- V6R1: New CL commands were introduced for key management, graphical configuration support via IBM Systems Director Navigator for i5/OS



Master Keys

- Eight master keys can be created that are securely stored in the Licensed Internal Code (LIC) with V5R4 and higher
  - Several versions of a single master key can exist
    - Up to 3 versions in V5R4
    - Up to 4 versions in V6R1
  - Master keys cannot be displayed
  - Master keys cannot be saved with V5R4, with V6R1 a new save function has been introduced
- New master keys in V6R1 for:
  - Save/Restore operation of master keys
  - ASP encryption

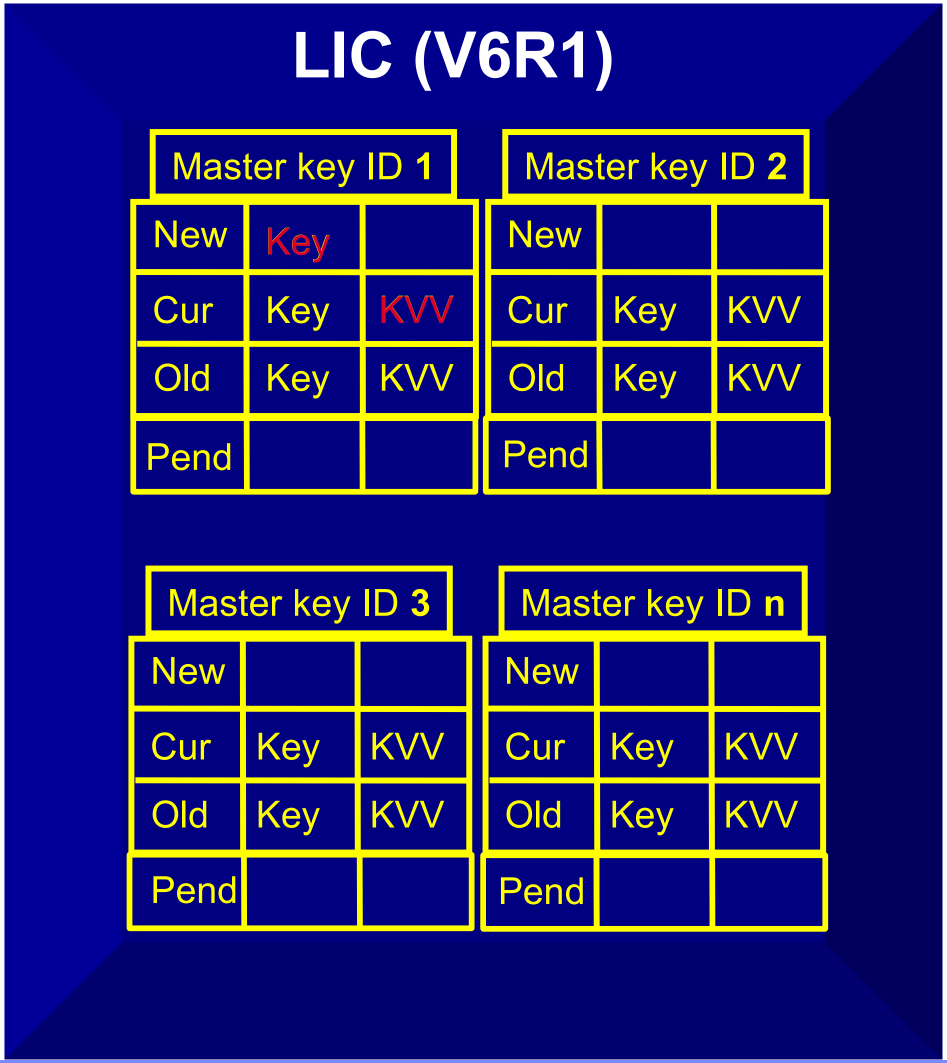
Master key management

- Creating a master key

ADDMSTPART CL cmd
Key
↓
SETMSTKEY CL cmd

- Changing a master key

ADDMSTPART CL cmd
Key
↓
SETMSTKEY CL cmd



Master key ID **1**		
New	Key	
Cur	Key	KVV
Old	Key	KVV
Pend		

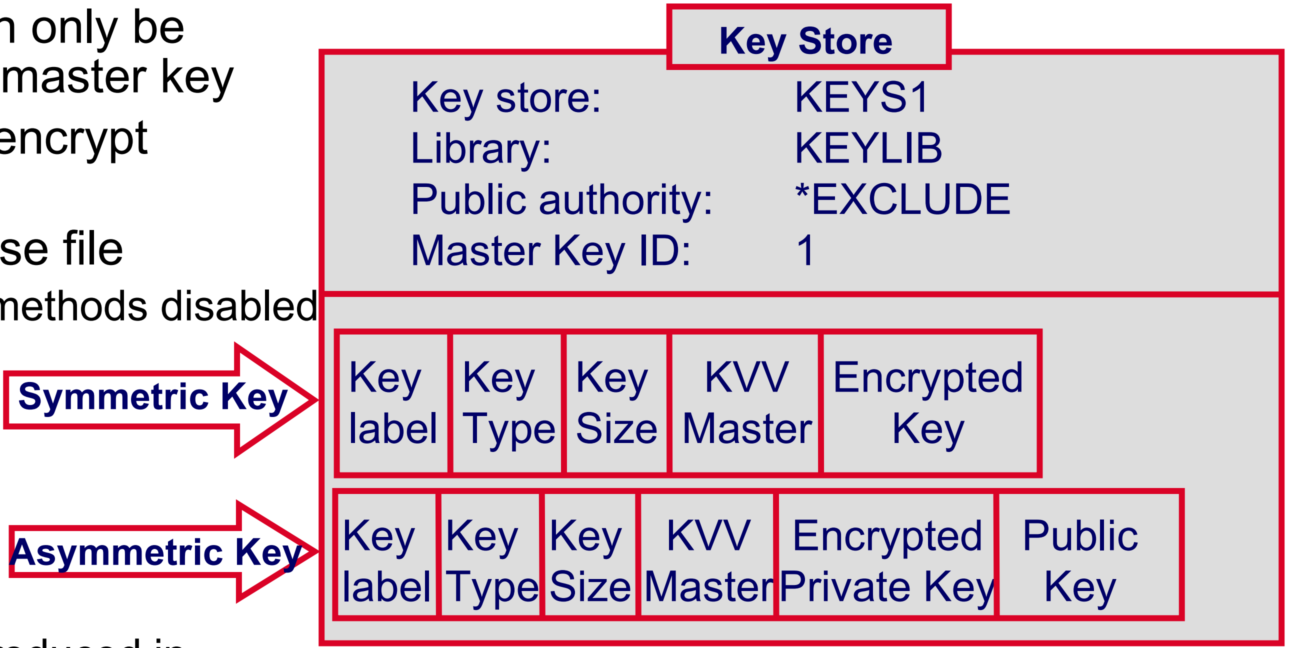
Master key ID **2**		
New		
Cur	Key	KVV
Old	Key	KVV
Pend		

Master key ID **3**		
New		
Cur	Key	KVV
Old	Key	KVV
Pend		

Master key ID **n**		
New		
Cur	Key	KVV
Old	Key	KVV
Pend		

Cryptographic Key Management – Key Store Files

- Cryptographic Services APIs (V5R4) as well as CL commands and the IBM Systems Director Navigator for i5/OS /V6R1) can be used to manage key stores
  - A single key store can only be encrypted under one master key
  - One master key can encrypt multiple key stores
  - Key store is a database file
    - •normal file access methods disabled

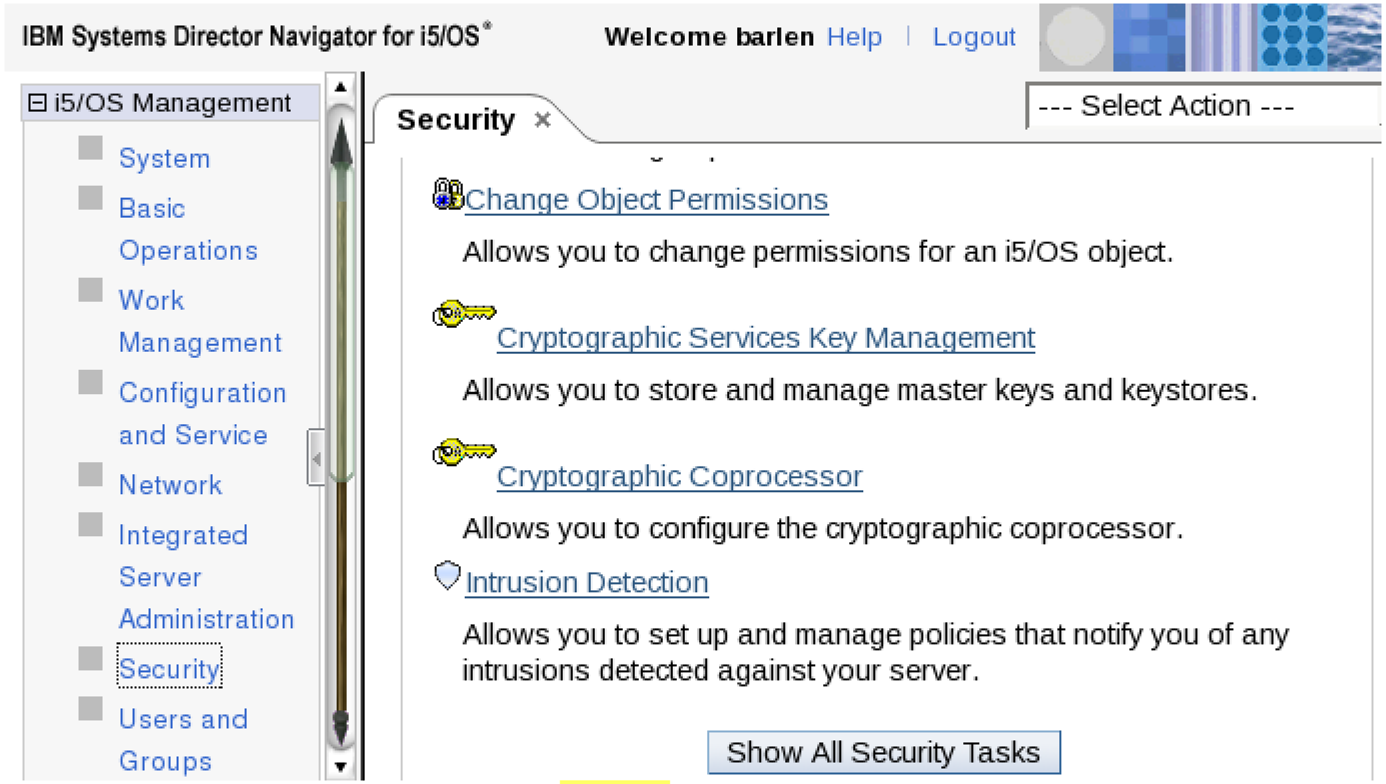

Key Store
Key store: KEYS1
Library: KEYLIB
Public authority: *EXCLUDE
Master Key ID: 1
Symmetric Key
Key label | Key Type | Key Size | KVV Master | Encrypted Key
Asymmetric Key
Key label | Key Type | Key Size | KVV Master | Encrypted Private Key | Public Key

Following commands were introduced in V6R1 for cryptographic key management

Add Keystore File Entry	ADDCKMKSFE
Create Keystore File	CRTCKMKSF
Display Keystore File Entry	DSPCKMKSFE
Generate Keystore File Entry	GENCKMKSFE
Remove Keystore File Entry	RMVCKMKSFE
Translate Keystore File	TRNCKMKSF

Cryptographic Key Management (CKM) options

IBM Systems Director Navigator for i5/OS



IBM Systems Director Navigator for i5/OS

Welcome barlen Help | Logout

i5/OS Management
- System
- Basic Operations
- Work Management
- Configuration and Service
- Network
- Integrated Server Administration
- Security
- Users and Groups

Security ×

--- Select Action ---

Change Object Permissions

Allows you to change permissions for an i5/OS object.

Cryptographic Services Key Management

Allows you to store and manage master keys and keystores.

Cryptographic Coprocessor

Allows you to configure the cryptographic coprocessor.

Intrusion Detection

Allows you to set up and manage policies that notify you of any intrusions detected against your server.

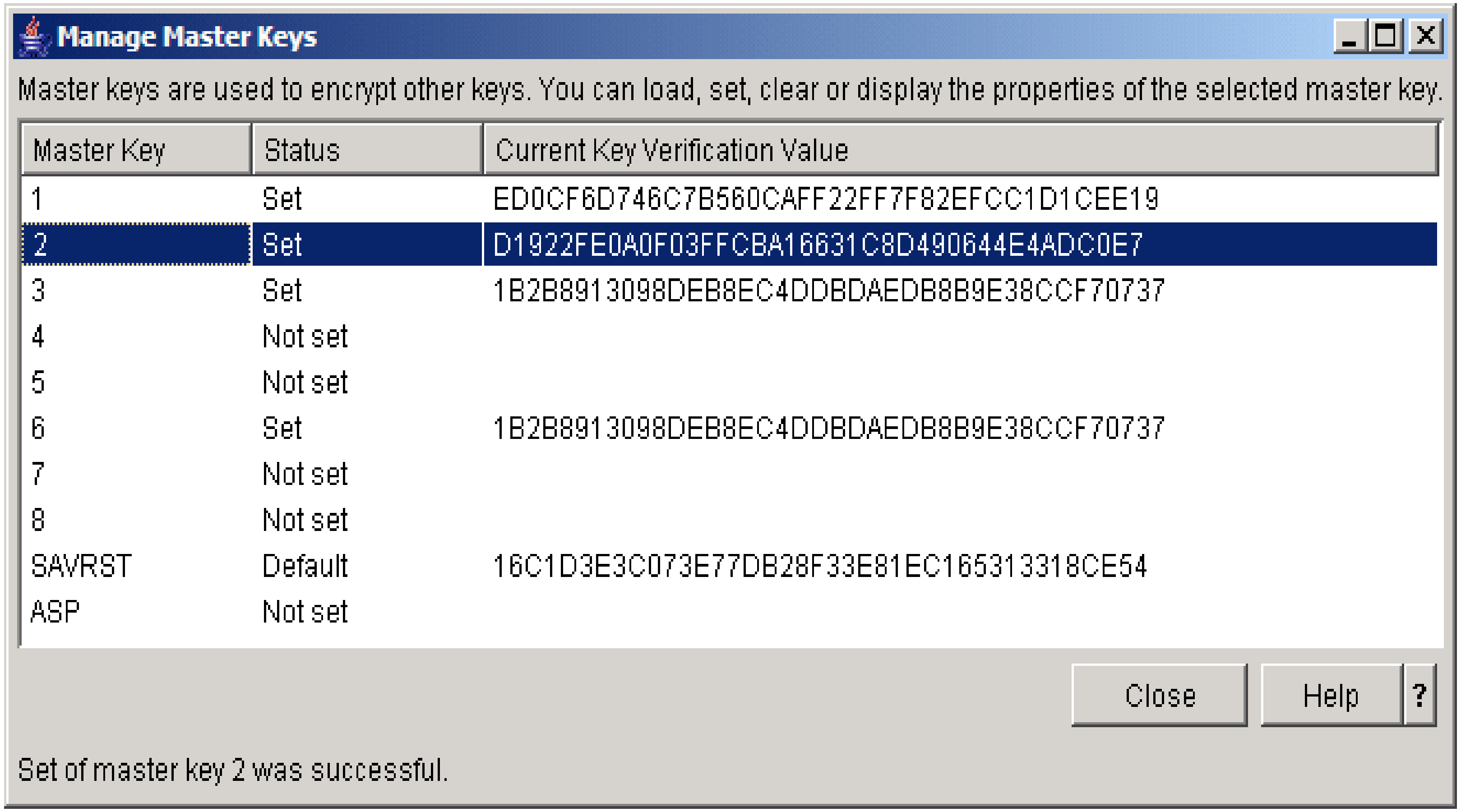
Show All Security Tasks

OR

CL Command (or APIs)

```
ADDMSTPART MSTKEY(1) PASSPHRASE() PASSLEN(*CALC)
```

Master Key Management



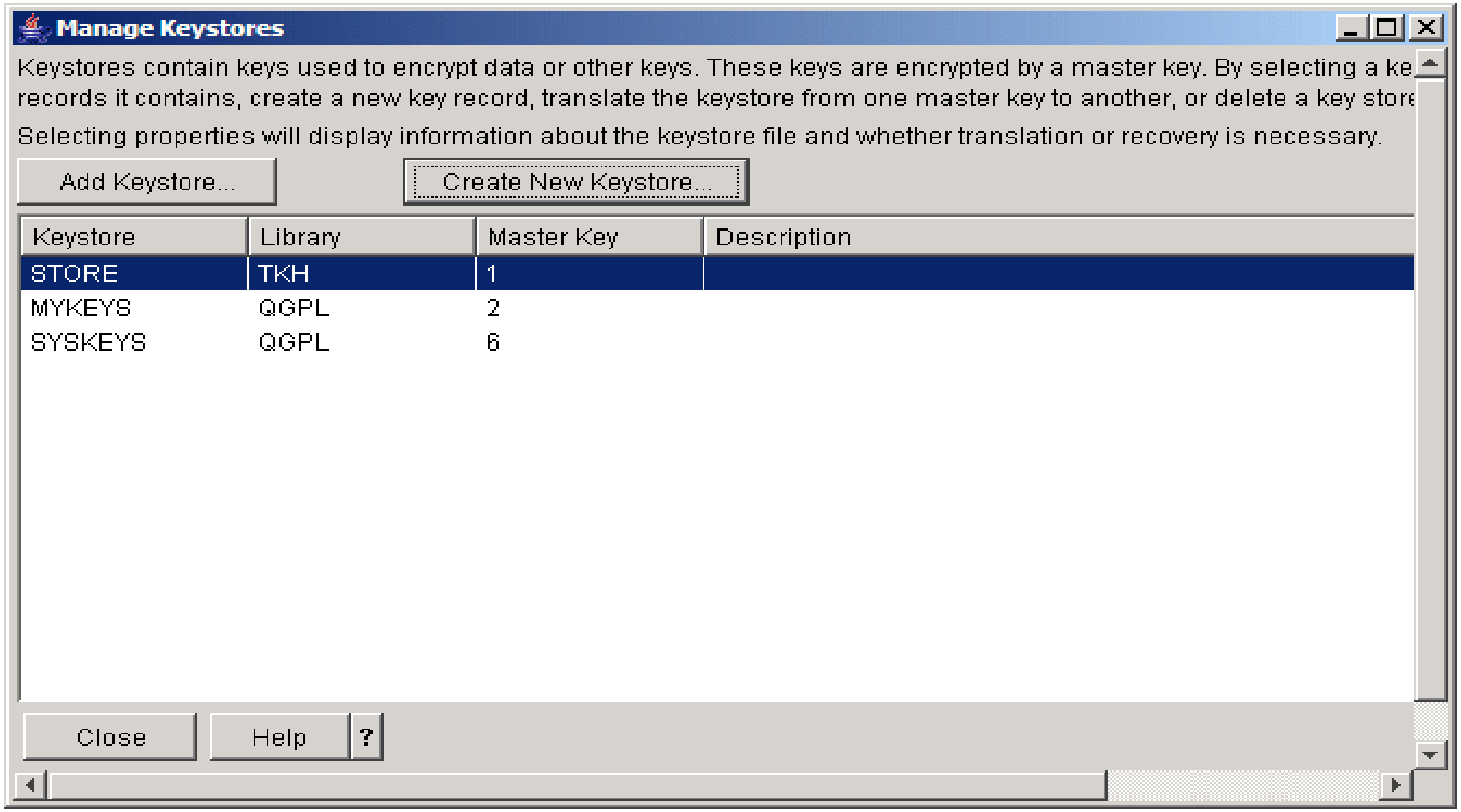
Manage Master Keys

Master keys are used to encrypt other keys. You can load, set, clear or display the properties of the selected master key.

Master Key	Status	Current Key Verification Value
1	Set	ED0CF6D746C7B560CAFF22FF7F82EFCC1D1CEE19
2	Set	D1922FE0A0F03FFCBA16631C8D490644E4ADC0E7
3	Set	1B2B8913098DEB8EC4DDBDAEDB8B9E38CCF70737
4	Not set	
5	Not set	
6	Set	1B2B8913098DEB8EC4DDBDAEDB8B9E38CCF70737
7	Not set	
8	Not set	
SAVRST	Default	16C1D3E3C073E77DB28F33E81EC165313318CE54
ASP	Not set	

Set of master key 2 was successful.

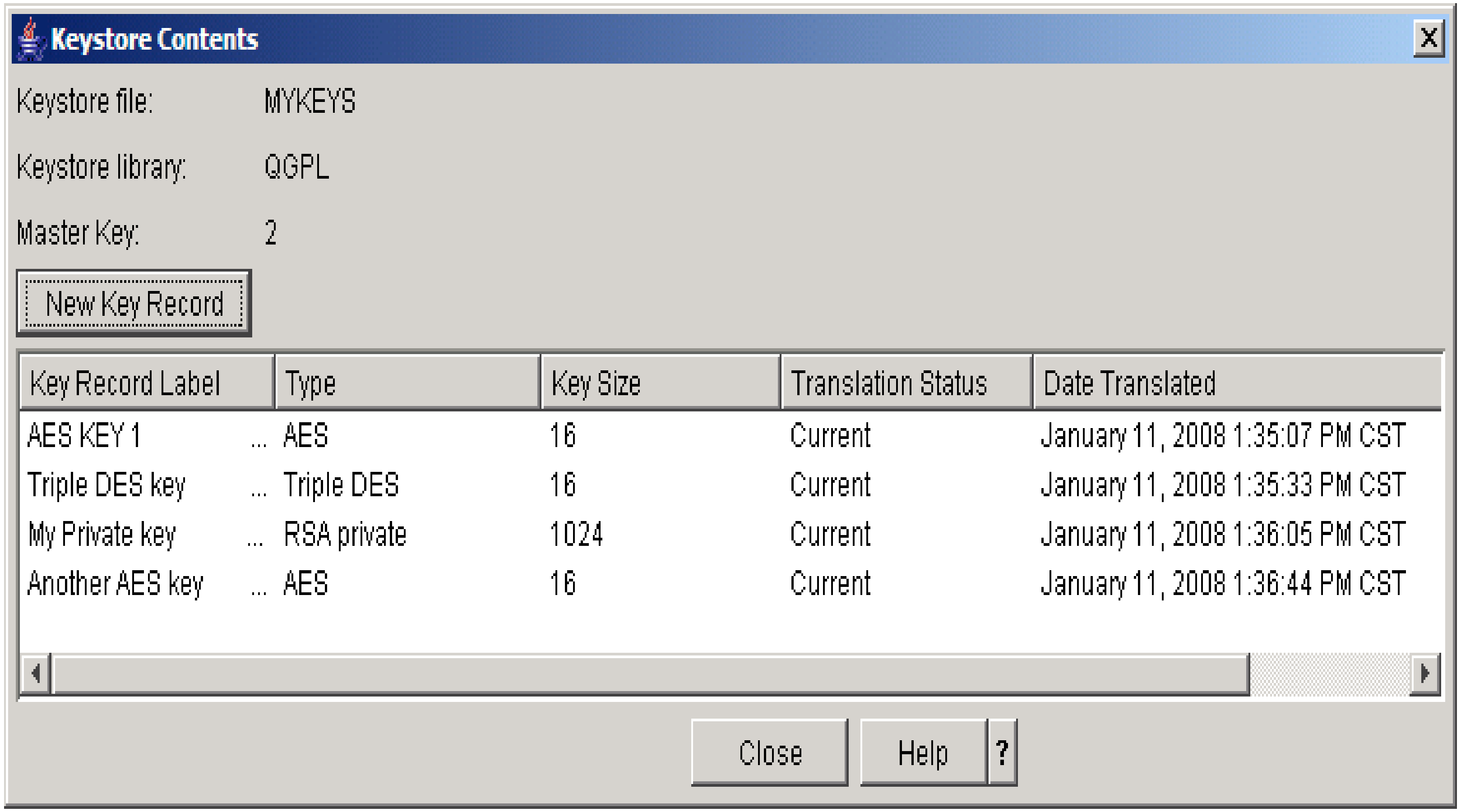
Keystore Management



Keystores contain keys used to encrypt data or other keys. These keys are encrypted by a master key. By selecting a ke
records it contains, create a new key record, translate the keystore from one master key to another, or delete a key store
Selecting properties will display information about the keystore file and whether translation or recovery is necessary.

Keystore	Library	Master Key	Description
STORE	TKH	1	
MYKEYS	QGPL	2	
SYSKEYS	QGPL	6	

Keystore Management

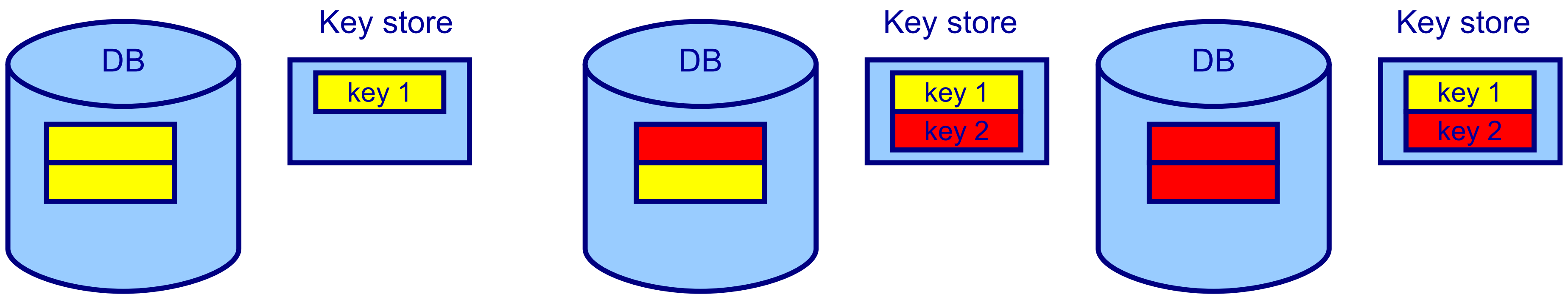


Keystore file:	MYKEYS
Keystore library:	QGPL
Master Key:	2

Key Record Label	Type	Key Size	Translation Status	Date Translated
AES KEY 1 ...	AES	16	Current	January 11, 2008 1:35:07 PM CST
Triple DES key ...	Triple DES	16	Current	January 11, 2008 1:35:33 PM CST
My Private key ...	RSA private	1024	Current	January 11, 2008 1:36:05 PM CST
Another AES key ...	AES	16	Current	January 11, 2008 1:36:44 PM CST

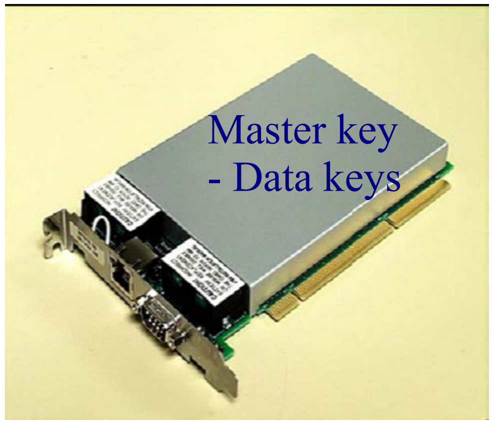
Key Changes and Backup

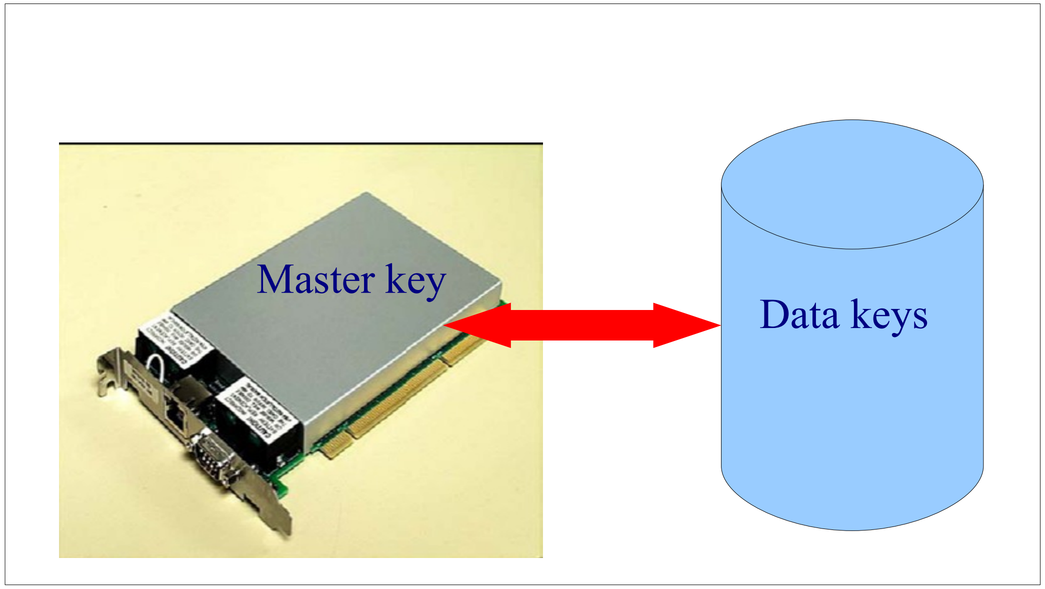
- Key encrypting keys should be changed on a regular basis
  - just like with passwords, encryption keys should be changed too
  - requires re-encryption of encrypted data with new key
  - independent of the chosen encryption function (DB2, crypto services, CCA), you should keep track of the different keys that are in use
- Exceptions might be, when encrypted data is stored offline along with the encrypted encryption key
  - you should then change the master key periodically
- Data backup and encryption key backup must be in sync, but should be kept in different places
  - may require multiple backup sets of encryption keys



Key management with the HSM

- The 4764 Cryptographic Coprocessor can be used on IBM Power Systems to perform cryptographic functions
- The coprocessor can also manage encryption keys
- A coprocessor's master key is used to protect (encrypt) the keys that are stored on the adapter card or in a key file external to the adapter card
  - in the latter case, only the adapter can access the keys





Overview of Network Encryption with IBM i (i5/OS)

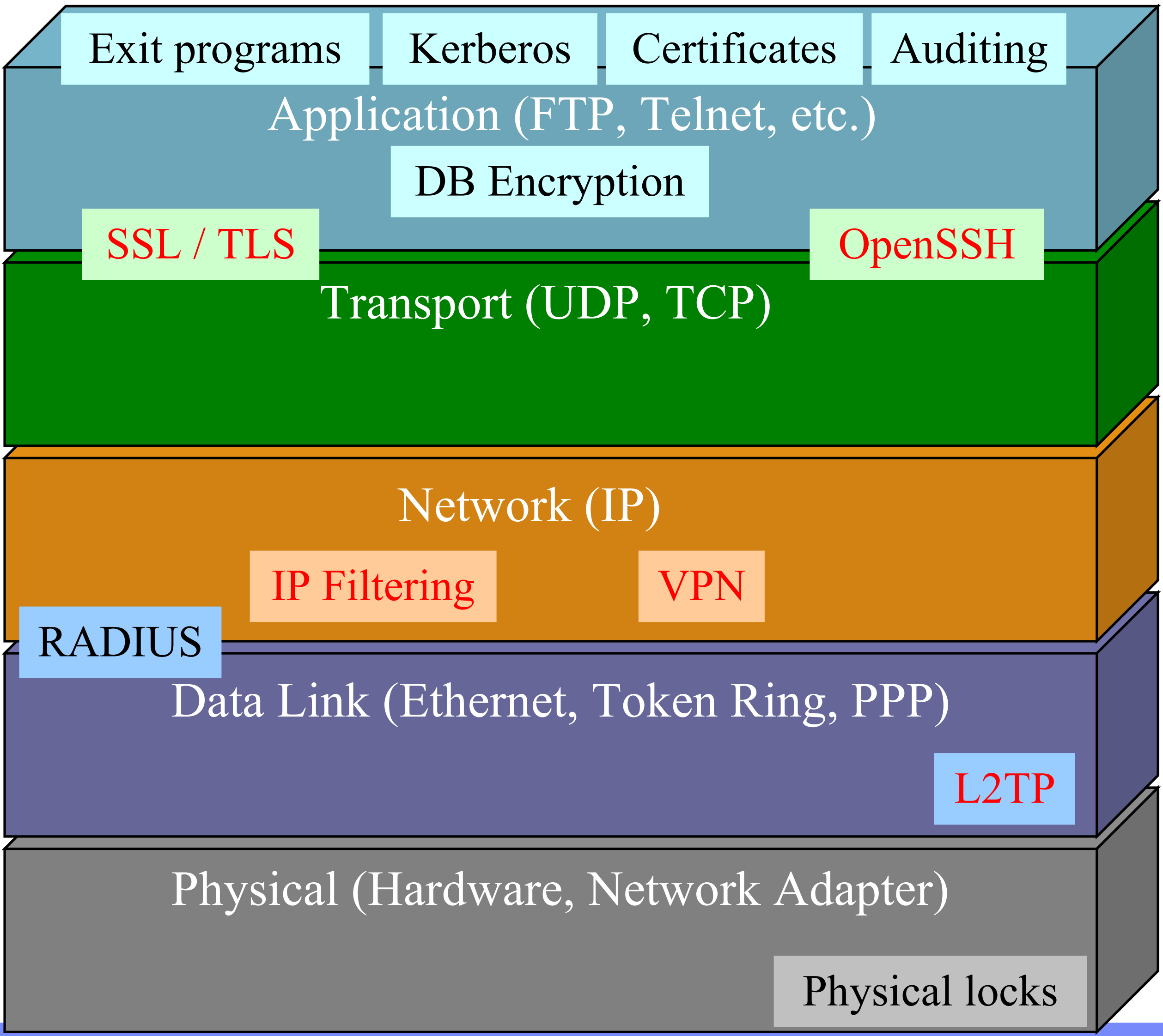
Introduction

- Several technologies are available in the market to secure data traffic across a network
- Data protection required to prevent sensitive data from being read by attackers
- By default, IP network traffic, such as Telnet or FTP transmit user / password information in the clear

```
25 R 88  23:29:33.252584              00096BAEABDC  00D0D32BFF44   ETHV2   Type: 0800
  Frame Type :  IP  DSCP: 0   ECN: 00-NECT  Length: 88 Protocol: TCP Datagram ID: 82C8
    Src Addr: 172.17.1.5   Dest Addr: 172.17.17.40  Fragment Flags: DON'T,LAST
  IP Header  :  4500005882C840003F064E88AC110105AC111128
  IP Options :  NONE
  TCP  . . . :  Src Port: 53358,Unassigned    Dest Port:    23,TELNET
   SEQ Number:  1717623935 ('6660DC7F'X)  ACK Number: 2957957869 ('B04EDAED'X)
               Code Bits: ACK PSH             Window:  7496  TCP Option: NO OP
  TCP Header :  D06E00176660DC7FB04EDAED80181D486FA100000101080A40479A7B5943F784
  Data:002212A000000 -……. . -1D9D3C5D5F2110735C2C1D5F5D5C1*............1...BARLEN2...BAN5NA*
       D5C5FFEF                                          *NA..                            *
```

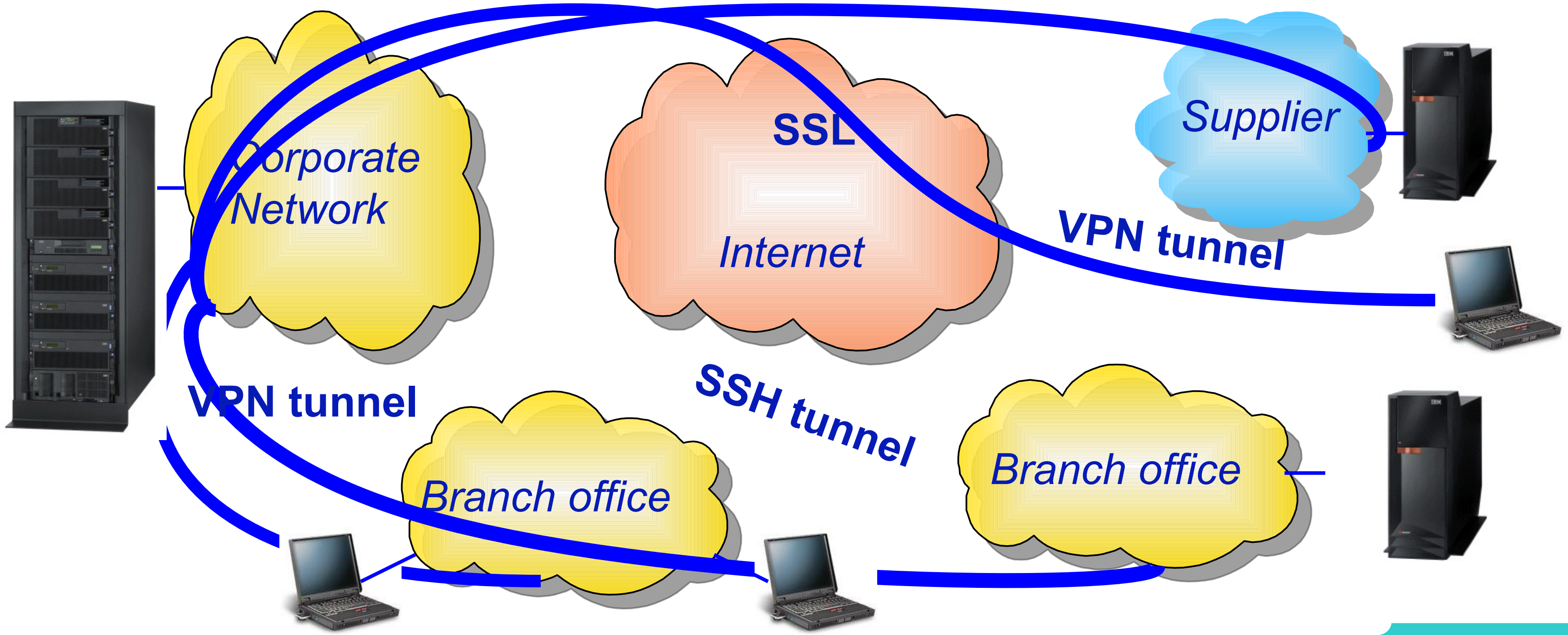
Security Features, Services, and Protocols

IBM i
offers security in
various layers!



Security when transmitting data over a network

- Secure data transmission with protocols that are built-in in IBM i (i5/OS)
- All inclusive, no additional cost



i5/OS provides:
- Secure Sockets Layer (SSL), Transport Layer Security (TLS)
- Virtual Private Networking (VPN) IPSec Protocol Standard
- Open Source security (Openssh, Openssl)

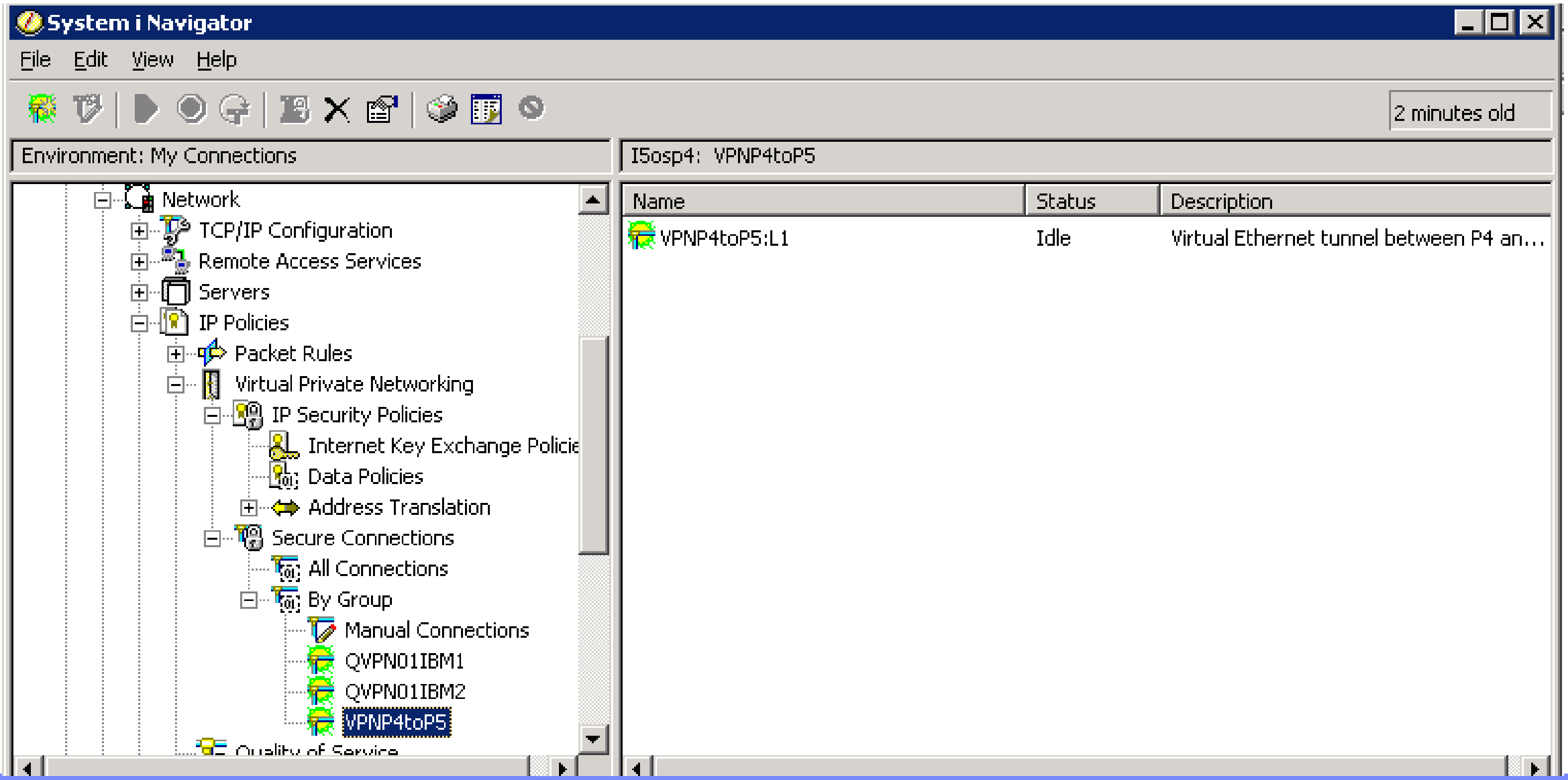
Authentication

Integrity

Confidentiality

Virtual Private Networking

- Available since OS/400 V4R4
- Built on standard IP Security (IPSec) framework
- Can be configured via System i Navigator
- When using RSA authentication (digital certificates), operating system option 34 (Digital Certificate Manager) and the HTTP server must be installed



Name	Status	Description
VPNP4toP5:L1	Idle	Virtual Ethernet tunnel between P4 an...

Virtual Private Networking

Confidentiality

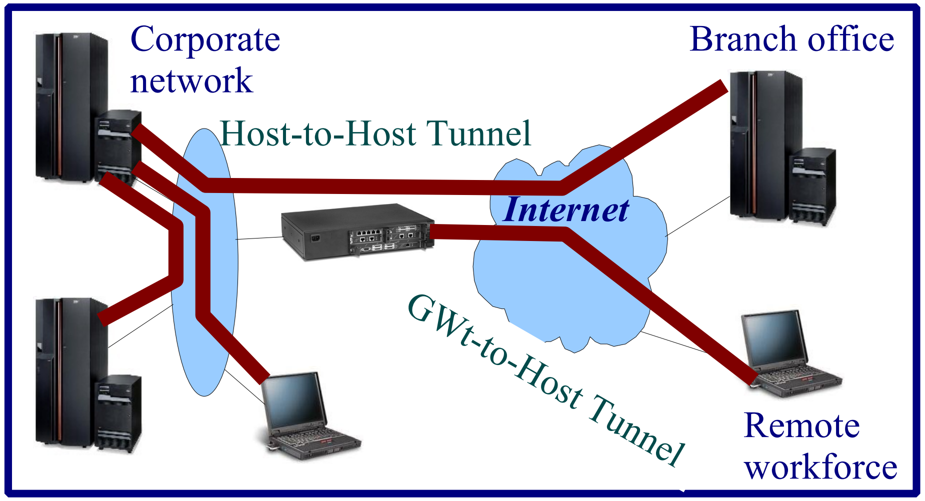
Integrity

Secure extension of your company's private intranet across a public network

VPN Requirements

- **Data Origin Authentication**
- **Data Integrity**
- **Data confidentiality**
- **Replay Protection**
- **Key Management**
- **Performance and Availability**
- **Interoperability**

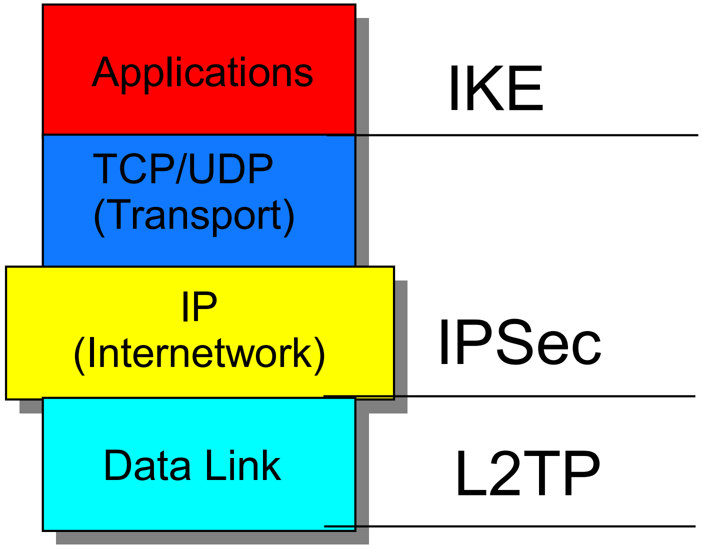
- Authenticate incoming data traffic
- Maintain data privacy
- Leverage ISP access locations
- Manage access as with private network



VPN Protocols

IP Security Architecture Protocols (IPSec)

- Open, standards-based, network layer security technology
- Supports authentication, integrity checking and encryption per packet
- Provides key management solution by using the Internet Key Exchange (IKE) protocols (used to be ISAKMP/Oakley)
- IETF standard in IPv6 -- optional in IPv4
- Used to secure L2TP tunnels



Layer 2 Tunneling Protocol (L2TP)

- Open, standards-based link layer technology
- Transports multiprotocol data over the Internet
- Cost-effective -- extends PPP connections to
- destination network
- IETF Internet draft, but emerging industry standard
- No inherent security features -- use IPSec for security

Secure Sockets Layer / Transport Layer Security

- Secure Socket Layer (SSL)
  - SSL V3.0 is the de facto industry standard and today is widely used in many applications to establish secure connections
  - SSL V3.0 was submitted as a draft to the Internet Engineering Task Force (IETF)
- The newer standard is called the Transport Layer Security (TLS) protocol
  - The Request for Comments for TLS is RFC2246
  - TLS Protocol Version 1.0
  - TLS V1.0 is supported in IBM i
- SSL and TLS will be interchangably in this presentation

Secure Sockets Layer / Transport Layer Security (2)

Authentication

Allows each communication partner to verify the identity of the other if required (normally the client verifies the server's identity)

Integrity

SSL/TLS ensures that data will not be changed while in transit

Authorization

At the application level based on client certificates identities provided over the secure session

Confidentiality

SSL/TLS primary responsibility is to encrypt the data. This encryption is actually done at the application layer

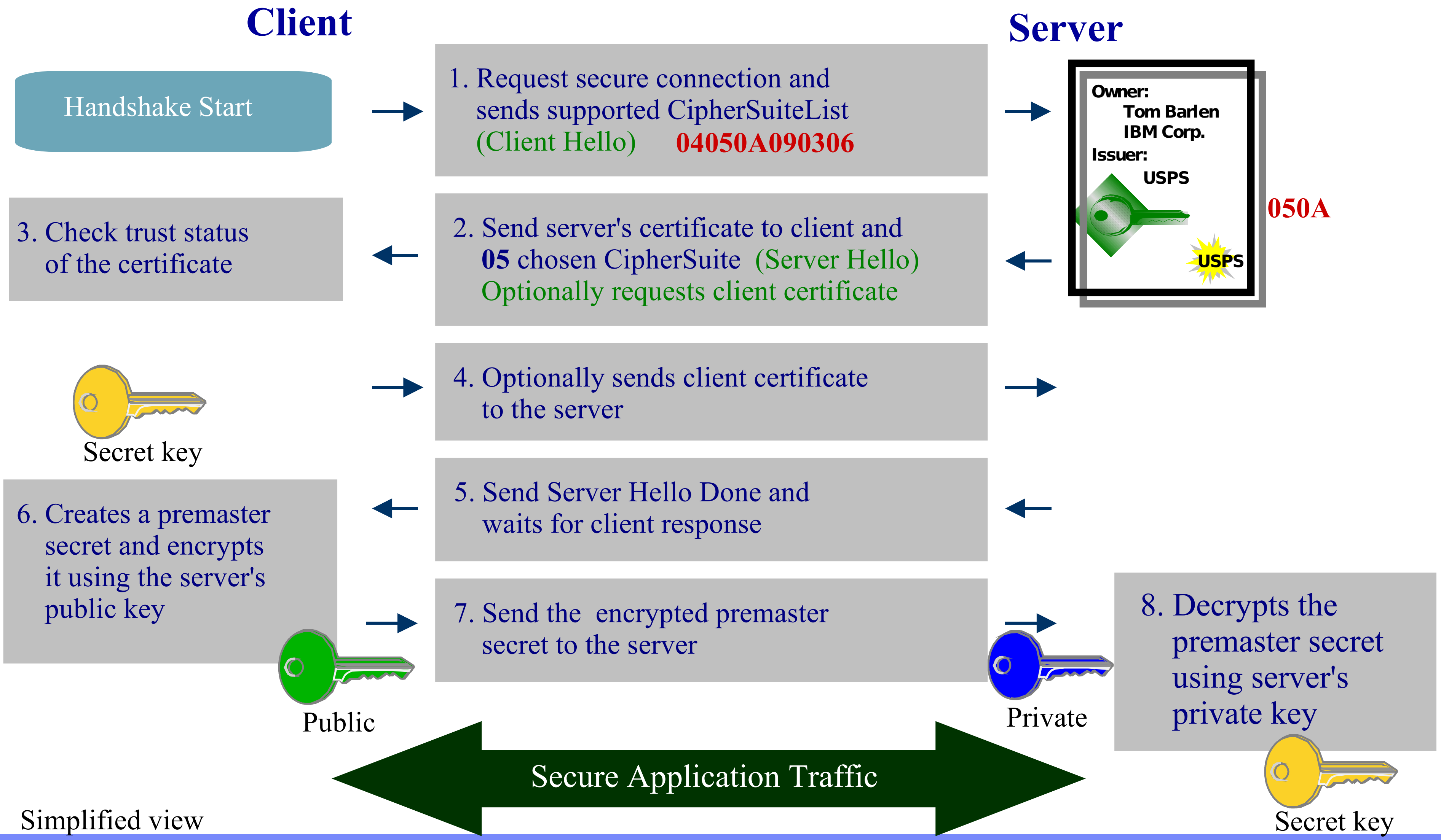
Audit/Logging

Application dependent, for example, HTTP server logs
Logging via exit programs

Secure Sockets Layer / Transport Layer Security (3)

- Implemented on top of the OSI Reference Model layer 4 (transport layer)
  - Applications must support SSL
  - Needs additional programming
  - Special sockets APIs
- SSL is not a single protocol. Instead, it consists of:
  - SSL record protocol
    - Sits on top of the transport layer and is used for encapsulation of various higher level protocols
  - SSL handshake protocol
    - Operates on top of the SSL record layer
    - Allows the client and server to authenticate each other
    - Negotiates an encryption algorithm and cryptographic keys before the application protocol receives or transmits data
    - Handshake involves digital certificates

SSL Handshake



Client

Server

Handshake Start

1. Request secure connection and sends supported CipherSuiteList (Client Hello) 04050A090306

2. Send server's certificate to client and **05** chosen CipherSuite (Server Hello) Optionally requests client certificate

3. Check trust status of the certificate

4. Optionally sends client certificate to the server

5. Send Server Hello Done and waits for client response

6. Creates a premaster secret and encrypts it using the server's public key

7. Send the encrypted premaster secret to the server

8. Decrypts the premaster secret using server's private key

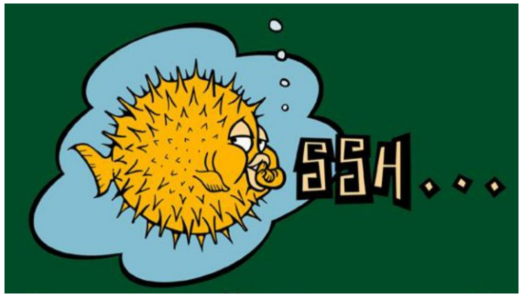
Secure Application Traffic

Simplified view

SSL/TLS Enabled IBM i Services

Service	SSL/TLS-enabled
Telnet Server	**Yes**
Telnet Client	**No**
FTP Server	**Yes**
FTP Client	**Yes**
HTTP Server	**Yes**
LDAP Server	**Yes**
LDAP Client	**Yes**
Host Servers	**Yes**
SMTP	**Yes (V6R1)**
POP	**Yes (V6R1)**
Java Toolbox	**Yes**

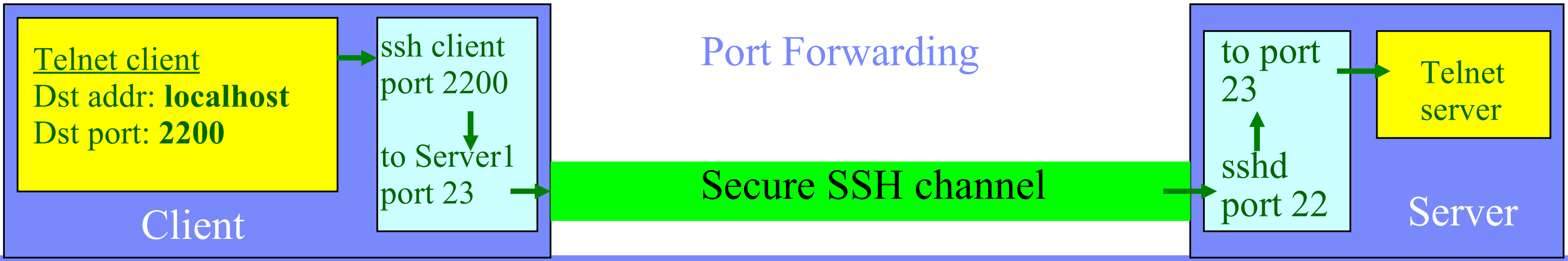
OpenSSH



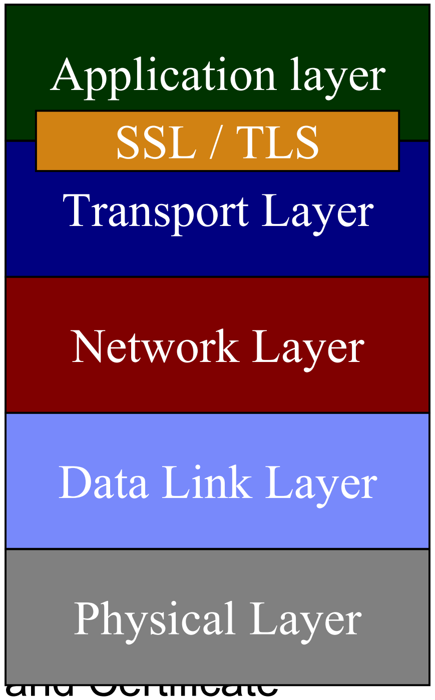
- Secure Shell (SSH) is a program to log into another computer over a network connection to run commands and copy files between computers
- Entire data traffic is encrypted including user and password information
- SSH is subject to licensing requirements
- OpenSSH is the free version of the SSH protocol suite
  - it does not use any patented components, such as the IDEA encryption algorithm
- Several utilities are available with OpenSSH
  - ssh - a secure command shell
  - sftp - a secure ftp alternative
  - scp - a secure file copy program
  - ssh-keygen - a public/private key pair generation and management tool
  - ssh-agent - an authentication agent that can store private keys
  - ssh-add - used to add private keys to a running ssh-agent
  - sshd - a daemon (server) program that handles incoming ssh connections
- Two protocols are available: SSH1 and SSH2

OpenSSH (cont‘d)

- OpenSSH also supports the following services and functions:
  - X11 forwarding
    - X11 forwarding allows the encryption of remote X windows traffic
  - Port Forwarding
    - Port forwarding allows forwarding of TCP/IP connections to a remote system over an encrypted channel
  - Data Compression
    - Uses zlib for compression
  - Kerberos and AFS Ticket Passing
    - Passes tickets for Kerberos and AFS on to the remote machine
  - Cryptographic functions
    - Uses the OpenSSL cryptographic library
- Information can be found at http://www.openssh.org



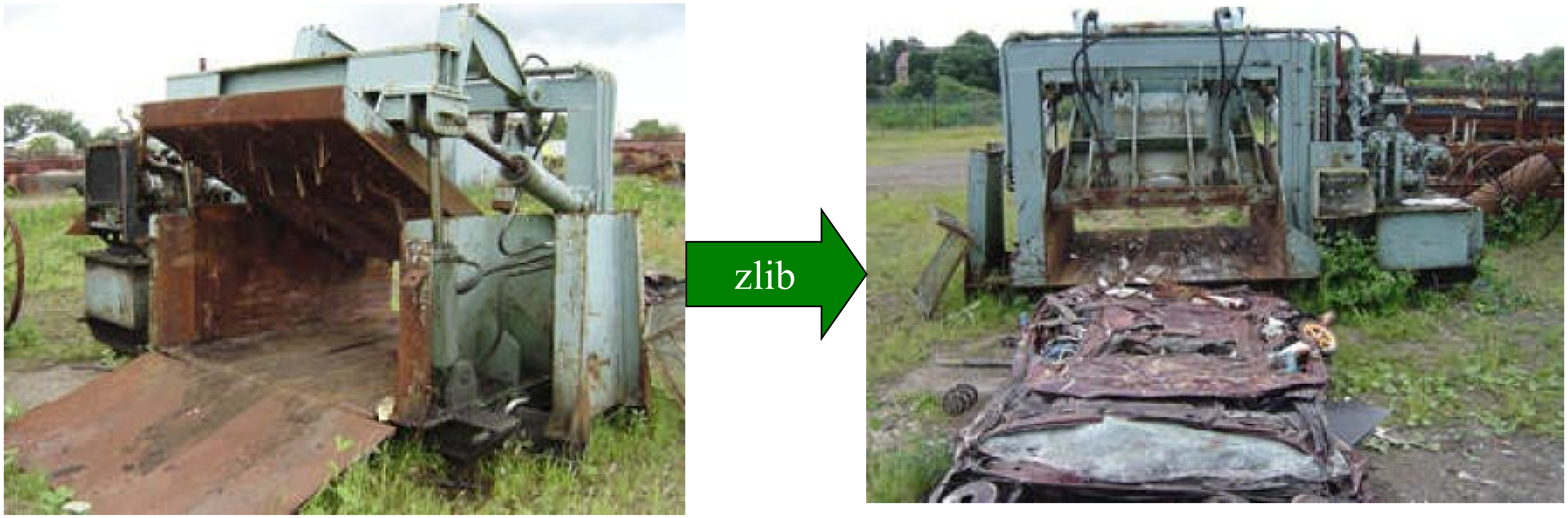
OpenSSL



- OpenSSL refers to an Open Source project that provides a full-featured SSL implementation
- It supports:
  - Secure Sockets Layer v2 and v3
  - Transport Layer Security v1
  - A general purpose cryptographic library
    Used by OpenSSH
- Allows programmers to write SSL/TLS sockets applications that can run on any platform that supports OpenSSL
- *openssl* command line tool can be used for:
  - Creation of RSA, DH and DSA key parameters
  - Creation of X.509 certificates, Certificate Signing Requests (CSRs) and Certificate Revocation Lists (CRLs)
  - Calculation of message digests
  - Encryption and decryption with ciphers
  - SSL/TLS client and server tests
  - Handling of S/MIME signed or encrypted mail
- Information is available from http://www.openssl.org

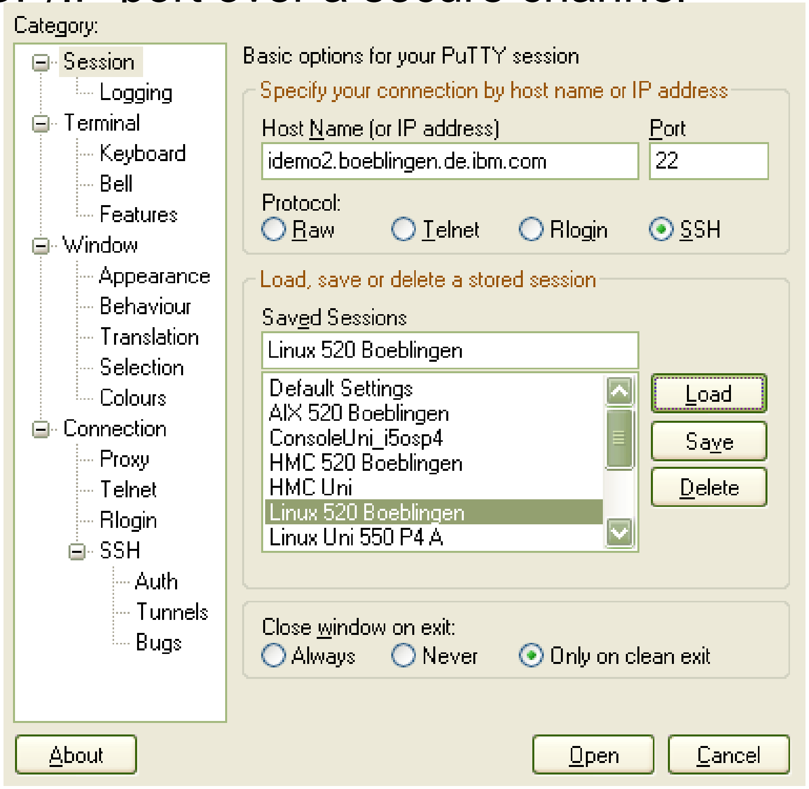
zlib

- zlib is a public compression algorithm that:
  - does not use patented material
  - is used in many compression products
  - can be freely downloaded and used for any purpose (personal or commercial)
  - however, is not the fastest algorithm available
- Information is available from http://www.gzip.org/zlib/



OpenSSH utilities - *ssh*

- *ssh* is the client utility used to connect to and run commands on a server running the SSH daemon (sshd)
  - basic syntax:
    - `ssh [user@]hostname [command]`
- ssh can also be used to forward an arbitrary TCP/IP port over a secure channel between the client and the server.
- The ssh client is also needed to programmatically connect to the Hardware Management Console (HMC) on the iSeries and System i5 and pSeries p5 models
- A popular ssh client is PuTTY
  - Available for Windows and Unix clients



Basic options for your PuTTY session

Specify your connection by host name or IP address

Host Name (or IP address): idemo2.boeblingen.de.ibm.com
Port: 22

Protocol: Raw Telnet Rlogin [✓ SSH]

Load, save or delete a stored session

Saved Sessions: Linux 520 Boeblingen

Default Settings
AIX 520 Boeblingen
ConsoleUni_i5osp4
HMC 520 Boeblingen
HMC Uni
Linux 520 Boeblingen
Linux Uni 550 P4 A

Close window on exit: Always Never [✓ Only on clean exit]

OpenSSH utilities – *sftp* and *scp*

- *sftp* is the client utility used to connect with and transfer files to or from a server running the SSH daemon (sshd)
  - basic syntax:
    - `sftp [user@]hostname [:file filename]`
- It is a completely different protocol than "normal" FTP
  - cannot use sftp to connect with an FTP server, or vice-versa
- *sftp* is similar to i5/OS FTP with some important differences:
  - sftp can only transfer data in binary format
  - sftp does not provide the enhanced functions available when transferring files in the QSYS.LIB (database) file system
  - sftp does not provide the CCSID data conversion options available with i5/OS FTP
  - supports compression (-C flag)

- *scp* is the client utility used to connect with and transfer a single file to or from a server running the SSH daemon (sshd)
  - basic syntax:
    - `scp [[user@]hostname] file1 file2`
- *scp* is similar to the sftp utility for transferring a single file

Portable Utilities for i5/OS – OpenSSH Implementation

- Installation via Restore License Program
  - `RSTLICPGM LICPGM(5733SC1) DEV(OPTxx) OPTION(*BASE) RSTOBJ(*ALL) LNG(2924)`
  - `RSTLICPGM LICPGM(5733SC1) DEV(OPTxx) OPTION(1) RSTOBJ(*PGM)`

- Product is installed under `/QOpenSys/QIBM/ProdData/SC1`
  - Subdirectories for OpenSSL, OpenSSH, and zlib
  - Each subdirectory contains a product/version specific directory
    - example: `/QopenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1` **(V5R3 and V5R4)**
    - `/QopenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.8.1p1` **(V6R1)**

- Configuration files for the sshd daemon are stored in:
  - `/QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc`

- Configuration files for user-specifc files are stored in:
  - `~/.ssh` (~ represents the user's home directory. Example: /home/barlen/)

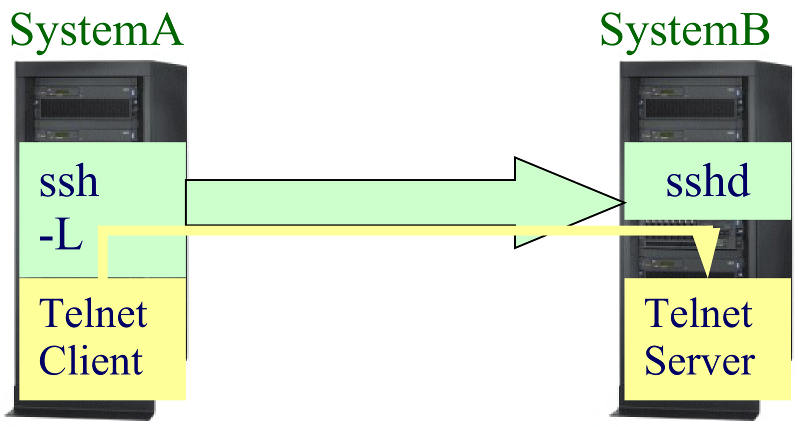
- No additional setup required for OpenSSL and zlib

Tunneling Telnet traffic in a secure ssh channel

- Tunneling or forwarding allows you to establish a secure connection for applications that would otherwise not protect data traffic
  - SMTP, POP, IMAP, SNMP, etc.
- Example: Setting up an ssh tunnel for Telnet between two iSeries servers

SystemA (manual tunnel setup)

1. `ssh -T -L5251:localhost:23 systemb`
   (keep the session open)
2. `TELNET RMTSYS(LOCALHOST) PORT(5251)`



SystemA (automatic tunnel setup)

1. The private key file is generated without passphrase

2.
   ```
   SBMJOB CMD(CALL PGM(QP2SHELL) PARM('/QOpenSys/usr/bin/ssh' '-T' '-N'
    '-L5251:localhost:23' 'systemb')) JOB(SSHTUN23) JOBQ(SSHTUNQ)
   ```
3. `TELNET RMTSYS(LOCALHOST) PORT(5251)`

SSL versus VPN versus OpenSSH

Feature	SSL	VPN	OpenSSH
Data confidentiality	Yes	Yes	Yes
Authentication	Server mandatory client optional	Host	User
Requires application support	Yes	No	No
Requires host support	No	Yes	No
Services	SSL-Enabled applications	All	All (binary file transfer)
Client configuration	Required for each application	Required for host	Required for each application
Tightly integrated into IBM i	Yes	Yes	No

Overview of Backup Encryption with IBM i (i5/OS)

i5/OS backup encryption options

- Two types of backup encryption
  - Hardware based
  - Software based
- Major difference is in performance and key management
- Backup encryption affects data stored on tape
- Software encryption not compatible with hardware encrypting tape drives

Hardware backup encryption options

- Two IBM tape drives exist that can be used for hardware-based backup encryption

1120 Tape Drives

LTO-4 Tape Drives

- Encryption is performed by dedicated hardware inside the tape unit
  - main CPU not involved in data encryption
- Data encrypted with AES 256-bit encryption
- Keys are managed outside the tape unit
  - Requires license program Encryption Key Manager (EKM)

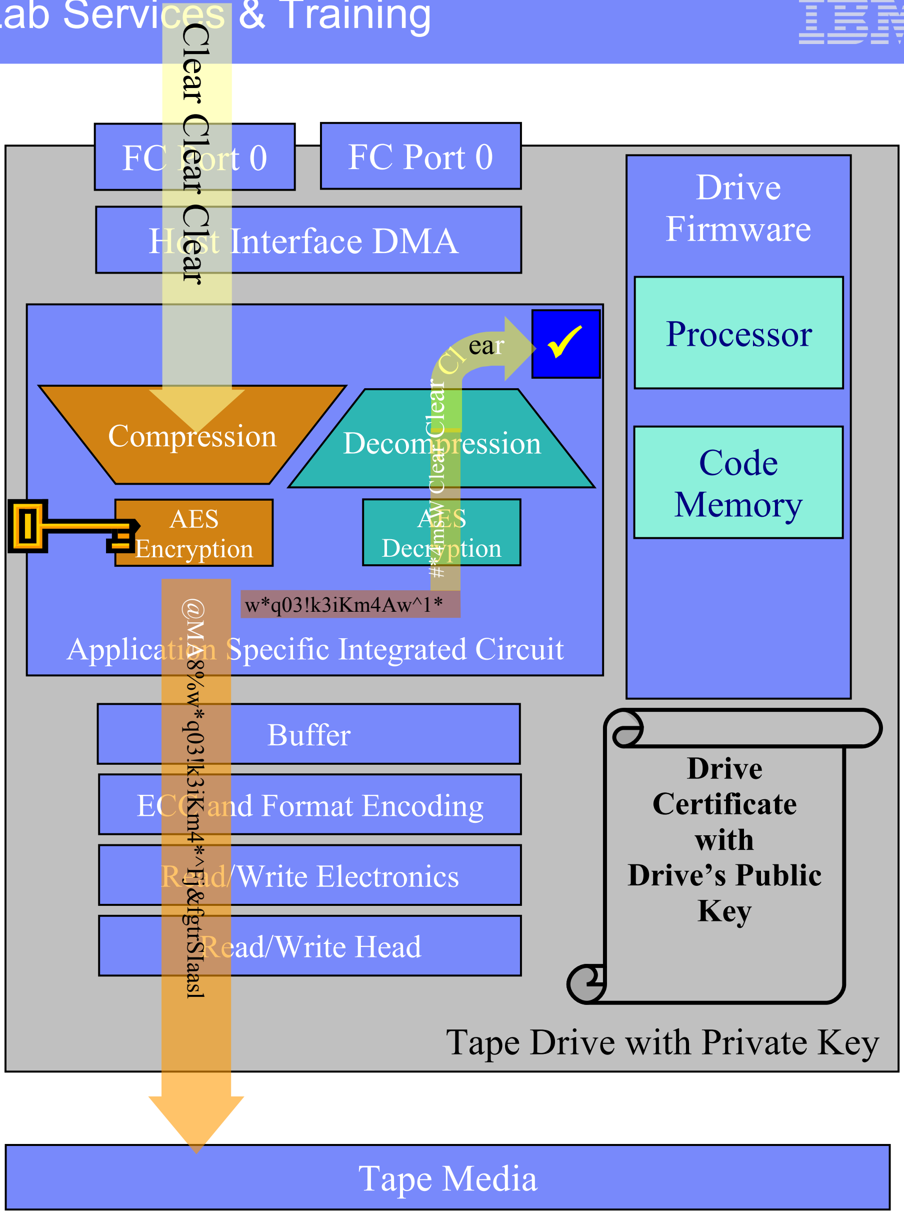
    - OR -
  - Tivoli Key Lifecycle Manager (TKLM)

IBM LTO-4 Encryption Overview

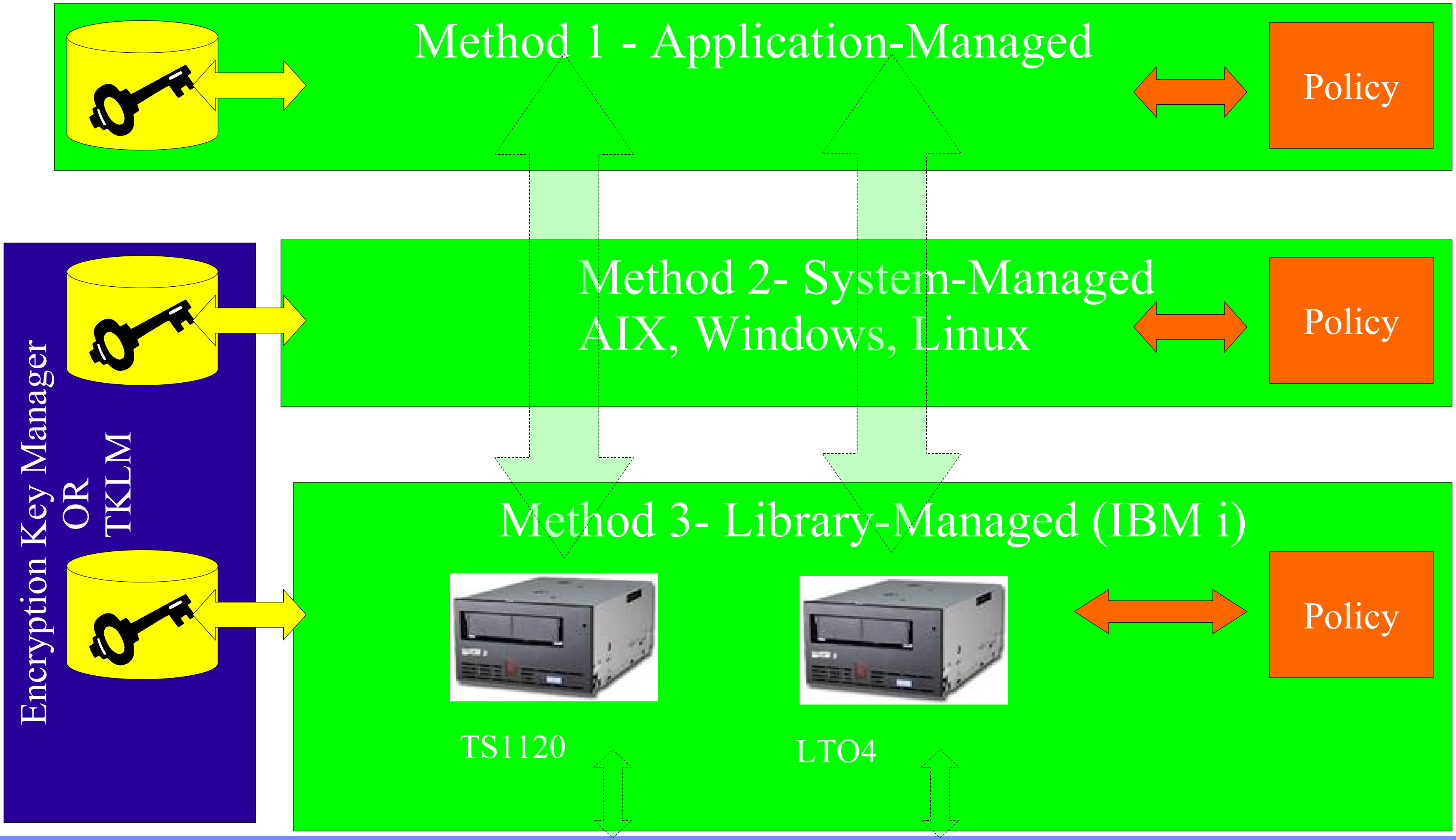
- Built on TS1120 Encryption Function
- Full encrypt/decrypt functions available on **SAS** and **FC** drives.
- SCSI Ultra160 drives are encryption media aware (i.e., will detect and appropriately handle encrypted cartridges) but are not capable of performing cryptographic operations.
- AES (256 bit per IEEE P1619.1) hardware based encryption of user data, with message authentication codes, after compression at effective line speed.
- On-the-fly checking of encrypted data through the hardware dataflow core.
- Supports the IBM Proprietary Protocol based application encryption operations. Same as TS1120.
- Supports T10 SSC-3 Standards based Security Protocol Commands
  - Security Protocol In (SPIN)
  - Security Protocol Out (SPOUT)
- Application transparent encryption
  - Library Managed Encryption (LME)
  - System Managed Encryption (SME)
    - Requires use of IBM Device Driver

TS1120 and LTO-4 Encryption

- Built-in AES 256-bit data encryption engine in every drive
- Look-aside decryption & decompression help assure data integrity.
- <1% performance impact on read/write throughput with encryption enabled & active
- Authentication: EKM queries drive certificate and uses public key to authenticate exchanges

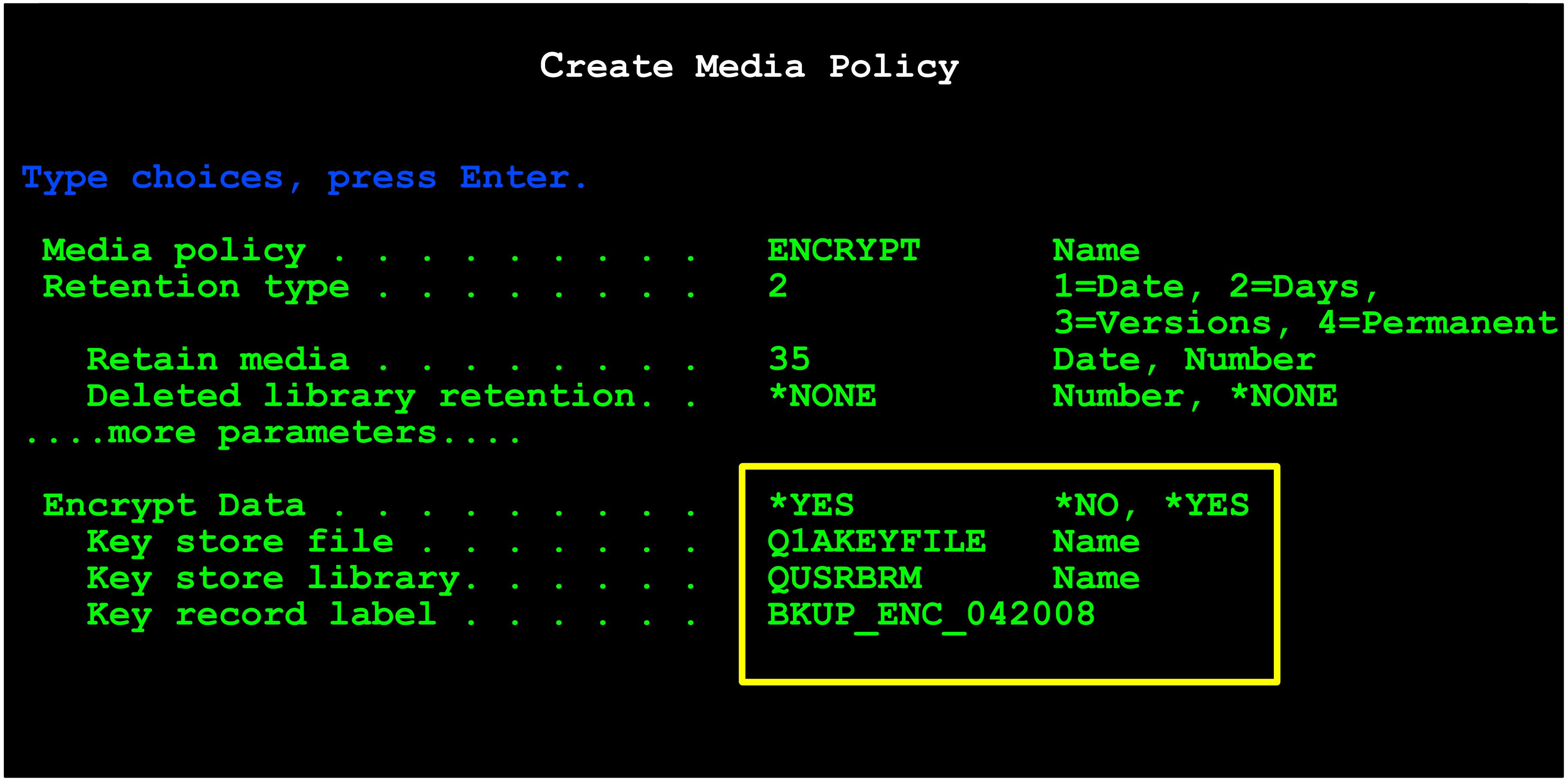


The solution supports three reference architectures for accessing keys and establishing policy.



Creating a BRMS media policy

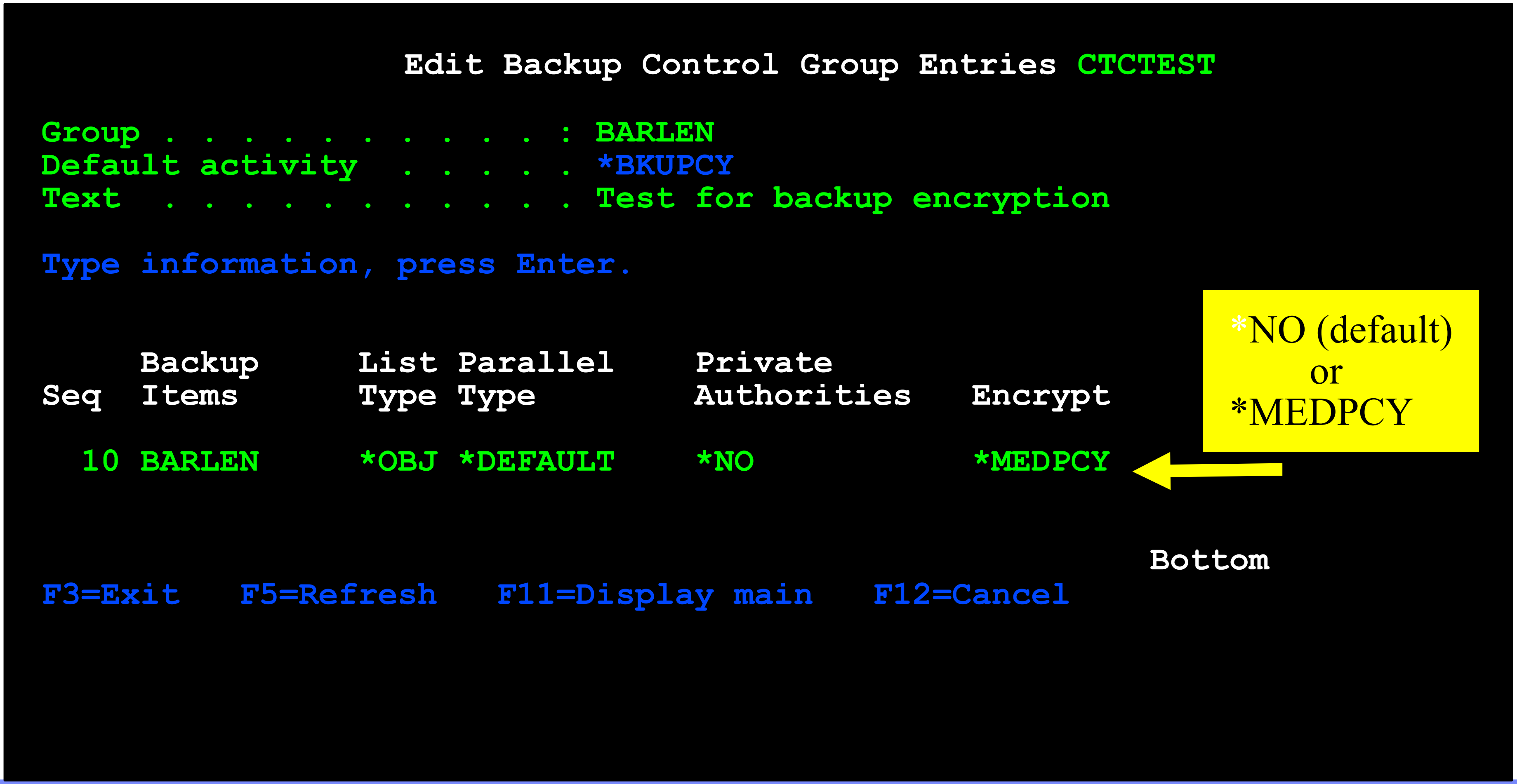
- Create a new or modify an existing media policy



```
                        Create Media Policy

Type choices, press Enter.

 Media policy . . . . . . . . .    ENCRYPT       Name
 Retention type . . . . . . . .    2             1=Date, 2=Days,
                                                 3=Versions, 4=Permanent
   Retain media . . . . . . . .    35            Date, Number
   Deleted library retention. .    *NONE         Number, *NONE
....more parameters....

 Encrypt Data . . . . . . . . .    *YES          *NO, *YES
   Key store file . . . . . . .    Q1AKEYFILE    Name
   Key store library. . . . . .    QUSRBRM       Name
   Key record label . . . . . .    BKUP_ENC_042008
```

Activating the media policy in a backup control group

- To encrypt data, the media policy must also be activated in a backup control group or archive control group



```
                    Edit Backup Control Group Entries CTCTEST

 Group . . . . . . . . . . . . . : BARLEN
 Default activity  . . . . . . . : *BKUPCY
 Text  . . . . . . . . . . . . . : Test for backup encryption

 Type information, press Enter.


      Backup          List Parallel     Private
 Seq  Items           Type Type         Authorities     Encrypt

   10 BARLEN          *OBJ *DEFAULT     *NO             *MEDPCY


                                                               Bottom
 F3=Exit   F5=Refresh   F11=Display main   F12=Cancel
```

*NO (default)
or
*MEDPCY

Overview of ASP Encryption with IBM i

Disk level encryption

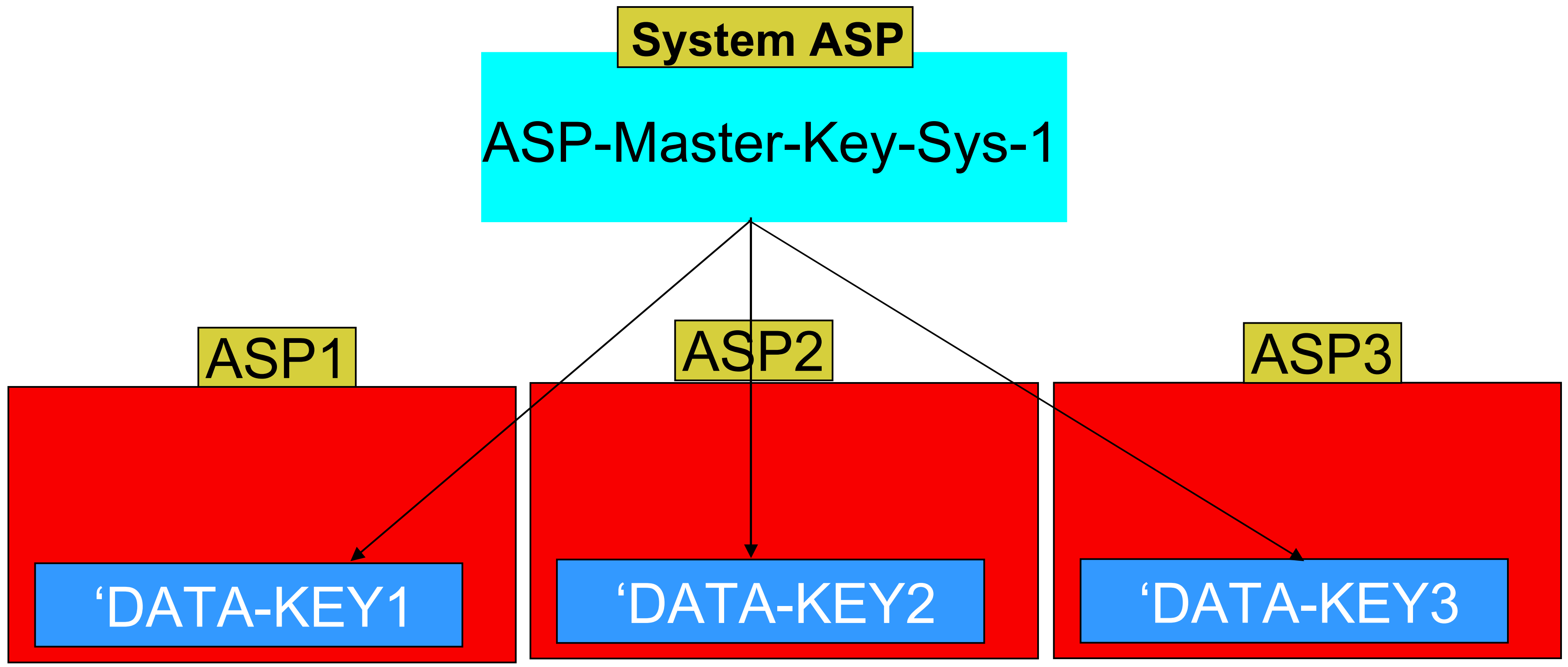
- Encryption of data at rest
  - Software solution
  - Minimal key management requirements

- Disk encryption protects data from a number of different threats:
  - Protects data transmission to and from the disk drive (important in a SAN environment).
  - Protects data transmission in the cross site mirroring environment (only when the data being mirrored is on an encrypted independent disk pool).
  - Protects data in the case of theft of the disk drive.
  - Protects data in the case of return or resale of a disk drive

Implementation approach

- Provide the capability to encrypt all data residing on an ASP
- Cryptographic keys will be stored in software but protected by “isolated” storage and master keys
- Minimal change required to an application
  - ASP level changes may be required
- 5761-SS1 Option 45 - Encrypted ASP Enablement is required
- Disk encryption cannot encrypt existing disk pools or independent disk pools
  - Only new pools can be encrypted
  - Once a pool is encrypted, encryption cannot be turned off

ASP Key Management

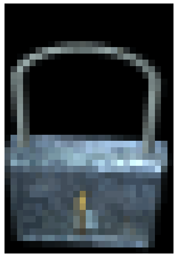


Overview of Database Encryption with IBM i (i5/OS)

Overview DB encryption

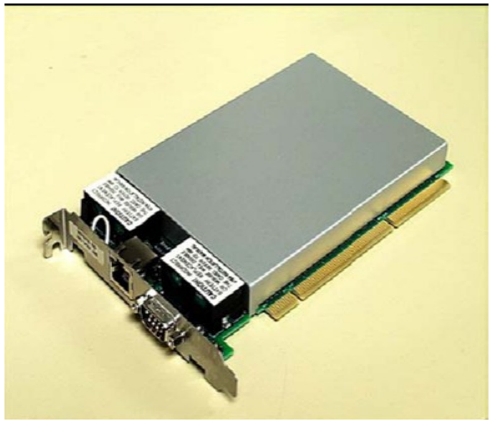
- Various cryptographic functions and services are available with IBM i (i5/OS)
  - Cryptographic Services APIs
  - Common Cryptographic Architecture (CCA) APIs
  - Cryptographic APIs within 5722-CR1 – Cryptographic Support for AS/400
    - Support has been withdrawn with release V6R1
  - DB2 built-in encryption/decryption services
  - Java IBMJCE and IBMJCEFIPS support
  - Machine Instruction (MI) calls
- Cryptographic hardware options are available to:
  - offload the main CPU from performing cryptographic functions
  - provide secure key store
  - provide higher security

Encryption / decryption is not performed automatically. It is the responsibility of the programmer to implement this service.



Hardware options (cont'd)

- PCI-X Cryptographic Coprocessor 4764 (#4806)
  - –Up to 4-5x performance of the predecessor PCI Coprocessor 4758-023 for most tasks
  - –Designed to meet FIPS 140-2 level 4 certification
  - –Supported functions
    - Generate random-numbers and MACs
    - Clone a master key securely
    - Support financial PIN-processing
    - Generate and validate digital signatures
    - Supports EMV 2000 (Europay/MasterCard/Visa) standard
    - Encrypt and decrypt data
    - Improve performance for SSL handshake processing (preferred for secure key store, but slower than 2058)
    - Import and export encrypted DES and Triple-DES keys securely
    - Secure key store
    - Supports EMV 2000 (Europay/MasterCard/Visa) standard
  - –Requires Cryptographic Device Manager (5733-CY1 for V5R3 and V5R4 / 5733-CY2 for V6R1)
    - Provides operating system for the coprocessor card
    - Linux-based
  - –IOP-less adapter feature for IBM i
  - –Minimum V5R3 and POWER5 server required



Cryptographic Services APIs

- Cryptographic Services APIs allow you to ensure:
  - Privacy of data
  - Integrity of data
  - Authentication of communicating parties
  - Non-repudiation of messages
  - Key management
- Introduced with i5/OS V5R3
  - some APIs also available for V5R2 via PTFs SI10060, SI10105, and MF31101
- All APIs can be processed on main CPU
  - Some APIs can leverage the 2058 cryptographic accelerator
  - Not all algorithms are supported on 2058

2058 has been withdrawn from marketing

Cryptographic Services APIs

- APIs can be used in any high level language
- Much simpler to use than CCA APIs

Key Generation APIs
Generate Symmetric Key
Generate PKA Key Pair
Generate Diffie-Hellman Parameters
Generate Diffie-Hellman Key Pair
Calculate Diffie-Hellman Secret Key

Key Management APIs (V5R4)
Clear Master Key
Create Key Store
Delete Key Record
Export Key
Extract Public Key
Generate Key Record
Import Key
Load Master Key Part
Retrieve Key Record Attributes
Set Master Key
Test Master Key
Translate Key Store
Write Key Record

Cryptographic Context APIs
Create Algorithm Context
Destroy Algorithm Context
Create Key Context
Destroy Key Context

Encryption / Decryption APIs
Encrypt Data
Decrypt Data
Translate Data

Authentication APIs
Calculate Signature
Verify Signature
Calculate MAC
Calculate Hash
Calculate HMAC

Pseudorandom Number Generation APIs
Generate Pseudorandom Numbers
Add Seed for Pseudorandom Number Generator

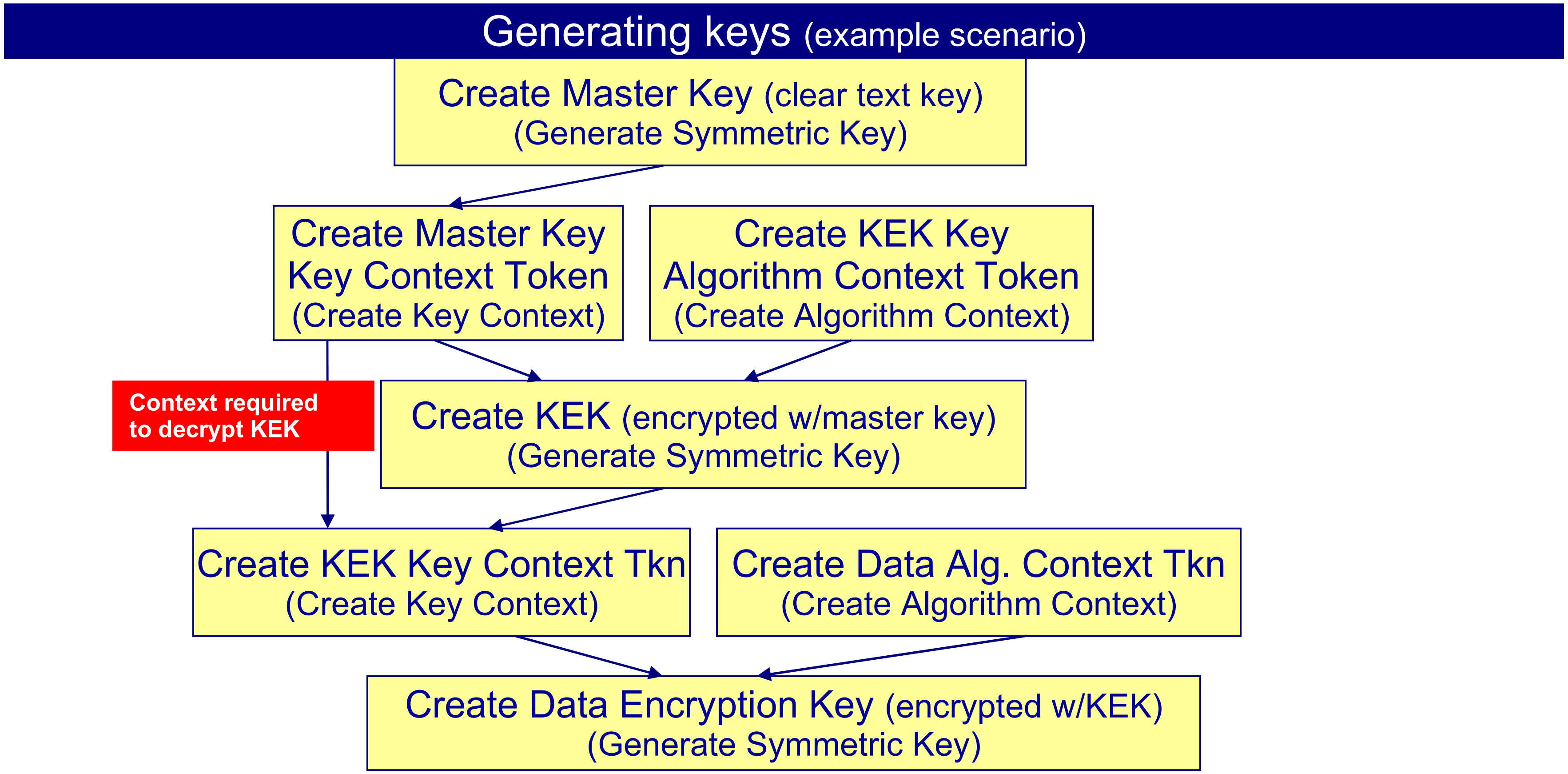
Cryptographic Services APIs Programming Environment

- APIs have input/output parameter
- RPG prototype definitions for API formats are available in:

  Library: QSYSINC File: QRPGLESRC Member: QC3CCI

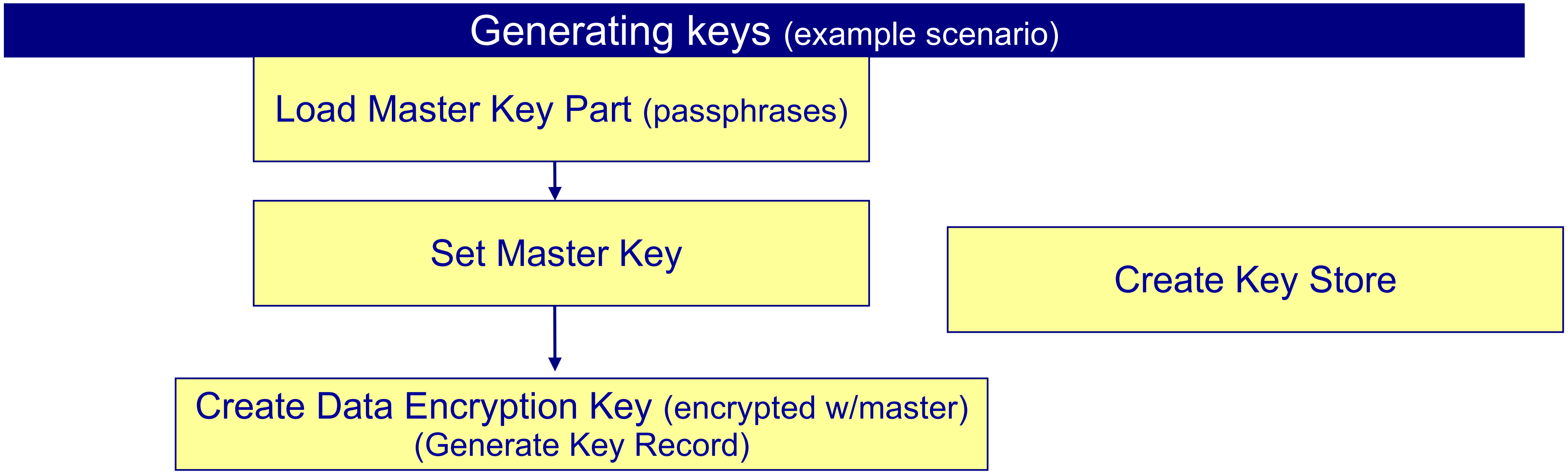
```
Encrypt Data (QC3ENCDT, Qc3EncryptData)


 Required Parameter Group:

1  Clear data                           Input   Char(*)    D clrData               1     const
2  Length of clear data                 Input   Binary(4)  D clrDataSize          10i 0  const
3  Clear data format name               Input   Char(8)    D clrDataFmt            8     const
4  Algorithm description                Input   Char(*)    D algDesc               1     const
5  Algorithm description format name    Input   Char(8)    D algDescFmt            8     const
6  Key description                      Input   Char(*)    D keyDesc               1     const
7  Key description format name          Input   Char(8)    D keyDescFmt            8     const
8  Cryptographic service provider       Input   Char(1)    D csp                   1     const
9  Cryptographic device name            Input   Char(10)   D cspDevNam            10     const options(*omit)
10 Encrypted data                       Output  Char(*)    D EncDta                1
11 Len of area provided for encr. dta   Input   Binary(4)  D DtaLenPrv            10i 0  const
12 Length of encrypted data returned    Output  Binary(4)  D DtaLenRtn            10i 0
13 Error code                           I/O     Char(*)    D errCod                1
```

Cryptographic Services APIs
Key Management Considerations prior V5R4


Generating keys (example scenario)
Create Master Key (clear text key)
(Generate Symmetric Key)
Create Master Key
Key Context Token
(Create Key Context)
Create KEK Key
Algorithm Context Token
(Create Algorithm Context)
Context required
to decrypt KEK
Create KEK (encrypted w/master key)
(Generate Symmetric Key)
Create KEK Key Context Tkn
(Create Key Context)
Create Data Alg. Context Tkn
(Create Algorithm Context)
Create Data Encryption Key (encrypted w/KEK)
(Generate Symmetric Key)

- Data encryption keys should be encrypted with key encrypting keys (KEKs)
- KEKs can be encrypted with a master key

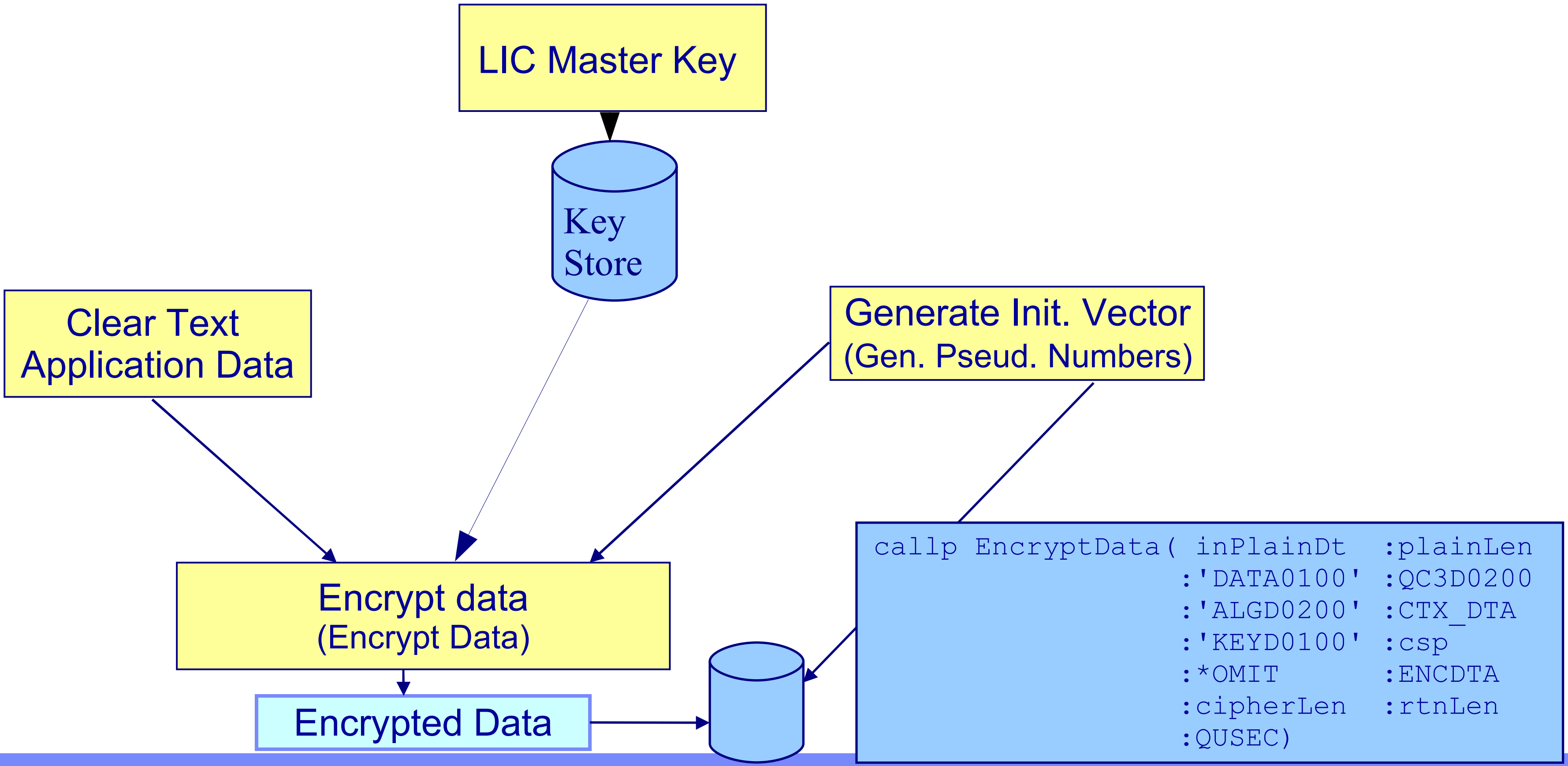
Cryptographic Services APIs
Key Management Considerations in V5R4


Generating keys (example scenario)
Load Master Key Part (passphrases)
Set Master Key
Create Key Store
Create Data Encryption Key (encrypted w/master)
(Generate Key Record)

- Data encryption keys could also be encrypted with key encrypting keys (KEKs). In this case, the KEK would have been encrypted under the master key, and the data key under the KEK

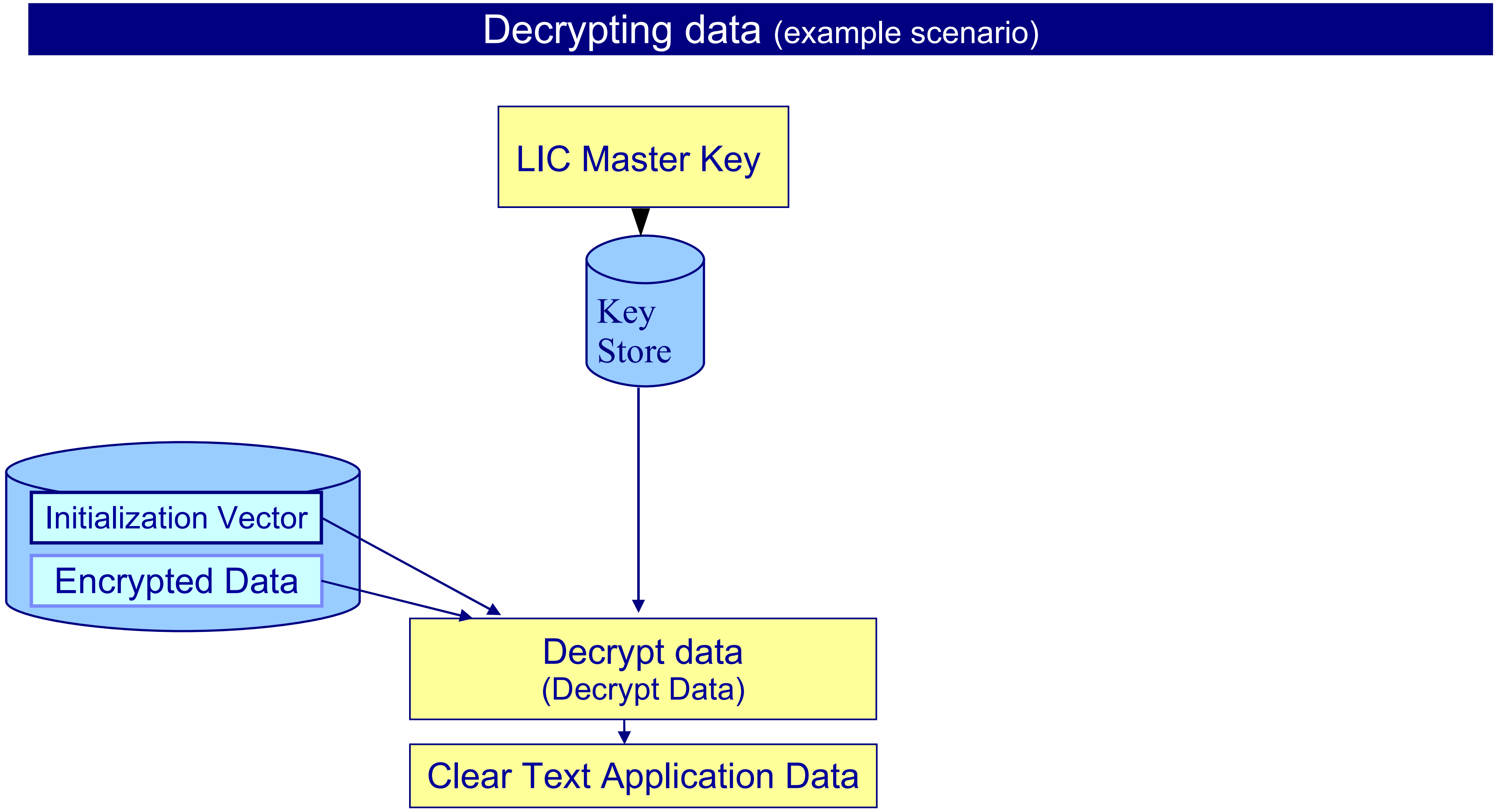
Cryptographic Services APIs Encryption

Encrypting data (example scenario)


LIC Master Key
Key Store
Clear Text Application Data
Generate Init. Vector (Gen. Pseud. Numbers)
Encrypt data (Encrypt Data)
Encrypted Data

```
callp EncryptData( inPlainDt   :plainLen
                  :'DATA0100'  :QC3D0200
                  :'ALGD0200'  :CTX_DTA
                  :'KEYD0100'  :csp
                  :*OMIT       :ENCDTA
                  :cipherLen   :rtnLen
                  :QUSEC)
```

Cryptographic Services APIs Decryption



Common Cryptographic Architecture (CCA) APIs

- CCA APIs are supported by the IBM Cryptographic Coprocessors 4758 and 4764
  - requires i5/OS option 5722-SS1 option 35 Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP)
- CCA CSP provides API support equivalent to the CCA Support Program that is available for z/OS, AIX, and Windows
- CCA CSP exploits the capabilities of the Cryptographic Coprocessors and allows you to do the following:
  - Create role-based access controls to define the level of access that you give to your users
  - Generate random-numbers
  - Clone a master key securely
  - Support financial PIN-processing
  - Generate and validate digital signatures
  - Encrypt and decrypt data
  - Protect keys
  - Import and export encrypted DES and Triple-DES keys securely
  - Generate Message Authentication Codes (MAC)

Notes:

- Common Cryptographic Architecture (CCA) APIs provide a variety of cryptographic processes and data security techniques. Your application program can call verbs to perform the following functions:
  - Encrypt and decrypt information, typically using the 3DES algorithm in the cipher block chaining (CBC) mode to enable data confidentiality
  - Hash data to obtain a digest, or process the data to obtain a message authentication code (MAC), that is useful in demonstrating data integrity
  - Create and validate digital signatures to demonstrate both data integrity and form the basis for non-repudiation
  - Generate, encrypt, translate, and verify finance industry personal identification numbers (PINs) and transaction validation codes with a comprehensive set of finance-industry-specific services
  - Manage the various DES and RSA keys necessary to perform the above operations
  - Control the initialization and operation of CCA
- The CCA API is designed so that a call can be issued from essentially any high-level programming language. The call, or request, is forwarded to the cryptographic-services access layer and receives a synchronous response; that is, your application program loses control until the access layer returns a response after processing your request. CCA APIs are used together with a cryptographic coprocessor, such as the 4758-023 or 4764.

Common Cryptographic Architecture (CCA) APIs

- Functions on the coprocessor are called verbs
- Located in QCCA library
- Coprocessor supported verbs and reference information can be found in the CCA Basic Services Reference and Guide

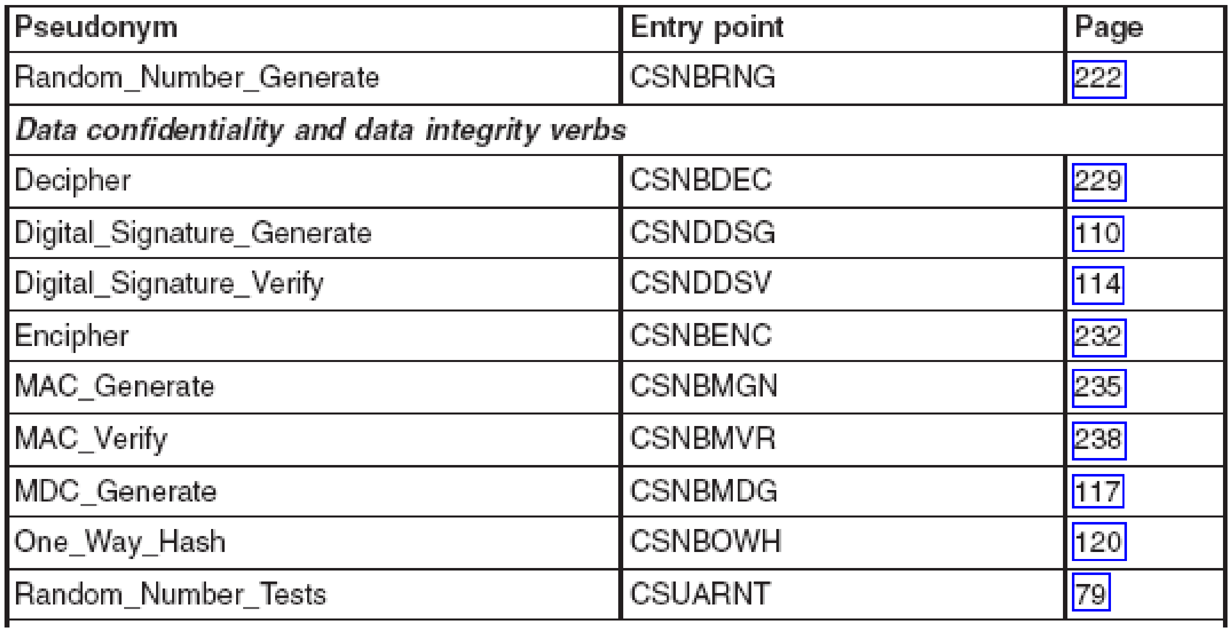


Table 80. Security API verbs in supported environments (continued)

Pseudonym	Entry point	Page
Random_Number_Generate	CSNBRNG	222
Data confidentiality and data integrity verbs		
Decipher	CSNBDEC	229
Digital_Signature_Generate	CSNDDSG	110
Digital_Signature_Verify	CSNDDSV	114
Encipher	CSNBENC	232
MAC_Generate	CSNBMGN	235
MAC_Verify	CSNBMVR	238
MDC_Generate	CSNBMDG	117
One_Way_Hash	CSNBOWH	120
Random_Number_Tests	CSUARNT	79

Common Cryptographic Architecture (CCA) APIs

- Coding examples in C and some in RPG available ☺

```
/* invoke verb to log on to the  card                        */

   CSUALCT( &return_code,
                     &reason_code,
                     &exit_data_length,
                     exit_data,
                     &rule_array_count,
                     (char *)rule_array,
                     profile,
                     &auth_parm_length,
                     auth_parm,
                     &auth_data_length,
                     argv[2]);

   if (return_code != OK)
   {
     printf("Log on failed with return/reason codes %ld/%ld\n\n",
     return_code, reason_code);
   }
```

```
C*********************************************
C* Call Logon Control SAPI
C****************************************
C        CALLP     CSUALCT       (RETURNCODE:
C                                 REASONCODE:
C                                 EXITDATALEN:
C                                 EXITDATA:
C                                 RULEARRAYCNT:
C                                 RULEARRAY:
C                                 USERID:
C                                 AUTHPARMLEN:
C                                 AUTHPARM:
C                                 AUTHDATALEN:
C                                 AUTHDATA)
C*-----------------------*
C* Check the return code *
C*-----------------------*
C      RETURNCODE    IFGT      0
```

DB2 Encryption

- DB2 UDB encryption supported since i5/OS V5R3
- Easier to use than cryptographic services or CCA APIs
- Allows encryption of data per column
  - the encryption key can be different for every row and column
- Supported through SQL interfaces only
  - application changes can be minimized with triggers & views
- RC2 block cipher encryption algorithm used (**V5R3**)
  - no other encryption algorithms can be used
  - key is derived by hashing a given password with the MD5 hashing algorithm
  - MD5 returns a 128 bit message digest
- 3DES algorithm support added with **V5R4**
  - password is hashed with a SHA-1 algorithm which returns a 128 bit message digest

Notes:

- DB2 built-in encryption functions were introduced with i5/OS V5R3. Encryption and decryption can be performed through standard SQL statements. The DB2 encryption allows you to encrypt data on a per column basis. The same or a different data encryption key can be used for different columns. With V5R3, the RC2 block cipher algorithm is used for encryption. Other encryption algorithms are not supported with V5R3. The encryption key is derived by hashing a specified password with the MD5 hashing algorithm. In V5R4, support for 3DES was added. The internal encryption algorithm used is Triple DES block cipher with padding, the 128 bit secret key is derived from the password using a SHA1 message digest.

DB2 Encryption
Encrypted Data Column Properties

- Depending on the current (non-encrypted) column properties, it is very likely that the column properties for encrypted data need to be changed
- The data type for the encrypted column must be one of the following:
  - BINARY, VARBINARY, CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, or BLOB
- Calculate the length of the encrypted data column

Without a password (key) hint specified	With a password (key) hint specified
number of bytes of input data string + 8 bytes (*1) + number of bytes to the next 8 byte boundary	number of bytes of input data string + 8 bytes (*1) + number of bytes to the next 8 byte boundary + 32 bytes for the hint length
Example: 8 bytes + 44 bytes data + 4 bytes = 56 bytes	Example: 8 bytes + 44 bytes data + 4 bytes + 32 bytes = 88 bytes

(*1) = 16 bytes if non-encrypted data string is a LOB or different CCSID values are used for the data string or the password

Notes:

- Even though the DB2 built-in encryption is simple to use, it usually cannot be used without implications on the column properties. The column length that will hold encrypted data need to be increased. If a password hint (label) is used, the column size must be even larger.
- When defining columns and distinct types to contain encrypted data:
  - The column must be defined with a data type of CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, BINARY, VARBINARY, or BLOB.
  - The length attribute of the column must include an additional *n* bytes, where *n* is the overhead necessary to encrypt the data as described above.
    - If a hint is not specified, n is 8 bytes (or 16 bytes if data-string is a LOB or different CCSID values are used for the data-string or the password-string) + the number of bytes to the next 8 byte boundary.
    - If a hint is specified, n is 8 bytes (or 16 bytes if data-string is a LOB or different CCSID values are used for the data-string, the password-string, or the hint-string) + the number of bytes to the next 8 byte boundary + 32 bytes for the hint length.
- Any assignment or cast to a column with a length shorter than the suggested data length may result in an assignment error or if the assignment is successful, a failure and lost data when the data is subsequently decrypted. Blanks are valid encrypted data values that may be truncated when stored in a column that is too short.
- Some sample column length calculations:

```
Maximum length of non-encrypted data           6 bytes
8 bytes                                        8 bytes (or 16 bytes)
Number of bytes to the next 8 byte boundary    2 bytes
                                               --------
Encrypted data column length                   16 bytes (or 32 bytes)

Maximum length of non-encrypted data           32 bytes
8 bytes                                        8 bytes (or 16 bytes)
Number of bytes to the next 8 byte boundary    8 bytes
                                               --------
Encrypted data column length                   48 bytes (or 56 bytes)
```

Administration of encrypted data: Encrypted data can only be decrypted on servers that support the decryption functions that correspond to the ENCRYPT_RC2 function. Hence, replication of columns with encrypted data should only be done to servers that support the decryption functions.

DB2 Encryption
Encrypting data

- Encryption is done with the ENCRYPT_RC2 or ENCRYPT_TDES function
  - ENCRYPT can also be used instead of ENCRYPT_RC2
  - Format: ENCRYPT(‘non-encrypted-data’, ‘password’, ‘hint’)
    - password and hint are optional, can also be set with the SET ENCRYPTION PASSWORD statement
  - Password should never be hardcoded, always pass the password as variables

```
INSERT INTO ord1
  VALUES(ENCRYPT_RC2('3750 999999 1111',:enckey1,:hintkey1),
  '8123412', 'John Doe')
```

or

```
SET ENCRYPTION PASSWORD :enckey1 WITH HINT :hintkey1
INSERT INTO ord1
  VALUES(ENCRYPT_RC2('3750 999999 1111'),
  '8123412', 'John Doe')
```

Notes:

- To encrypt data in a table, the ENCRYPT_RC2 function is used. This function can also be called via the ENCRYPT alias. In V5R4, the ENCRYPT_TDES function was added for Triple DES encryption.
- Example:
  - ENCRYPT('non-encrypted-data', 'password', 'hint')
- The ENCRYPT_RC2 function returns a value that is the result of encrypting data-string using the RC2 encryption algorithm. The password used for decryption is either the password-string value or the ENCRYPTION PASSWORD value (assigned by the SET ENCRYPTION PASSWORD statement). If the encryption facility is not available an error is returned.
- data-string
  - An expression that returns the string value to be encrypted. The string expression must be a string built-in data type. The length attribute for the data type of data-string must be less than the maximum length of the result data type minus n, where n is the amount of overhead necessary to encrypt the value.
    - If a hint-string is not specified, n is 9 bytes (or 17 bytes if data-string is a LOB or different CCSID values are used for the data-string or the password-string.)
    - If a hint-string is specified, n is 41 bytes (or 49 bytes if data-string is a LOB or different CCSID values are used for the data-string, the password-string, or the hint-string.)
- password-string
  - An expression that returns a character string value with at least 6 bytes and no more than 127 bytes. The expression must not be a CLOB. The value represents the password used to encrypt the data-string. If the value of the password argument is null or not provided, the data will be encrypted using the ENCRYPTION PASSWORD value, which must have been set using the SET ENCRYPTION PASSWORD statement.
- hint-string
  - An expression that returns a character string value with up to 32 bytes that will help data owners remember passwords (For example, 'Ocean' is a hint to remember 'Pacific'). If a hint value is specified, the hint is embedded into the result and can be retrieved using the GETHINT function. If this argument is the null value or is not provided, no hint will be embedded in the result.

DB2 Encryption Hints

- A hint is an expression that will help to remember the password that was used for encryption (i.e. a key label)
- It can also be used to obtain the correct password (key)
  - i.e. if different passwords are used to encrypt different columns in a single DB or columns in different databases
- It is optional to specify a hint during encryption
- You can determine the password hint with the GETHINT built-in function

```
SELECT GETHINT(cardno), name, type FROM ord1
```

```
Position to line  . . . . .
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8..
GETHINT ( CARDNO )                NAME                                 TYPE
tomkey                            Thomas Barlen                        Amex
markey                            Marion Barlen                        Visa
tomkey                            Kent Milligan                        Master
********  End of data  ********
```

Notes:

- A hint indicates that a value is specified that will help data owners remember passwords (for example, ‘Mycar' as a hint to remember ‘AudiA3'). If a hint value is specified, the hint is used as the default for encryption functions. The hint can subsequently be retrieved for an encrypted value using the GETHINT function. If this clause is not specified and a hint is not explicitly specified on the encryption function, no hint will be embedded in encrypted data result.

DB2 Encryption - Decrypting data

- DB2 decrypt functions can only decrypt data that were encrypted with the ENCRYPT_RC2 or ENCRYPT_TDES functions
- Different decryption built-in functions available for different input data types (DECRYPT_CHAR, DECRYPT_BIT, DECRYPT_BINARY, and DECRYPT_DB)
- Password can be set within the decrypt function or via the SET ENCRYPTION PASSWORD statement

```
SELECT DECRYPT_CHAR(cardno, :enckey1), name, type FROM ord1 WHERE
name = 'Thomas Barlen'
```

Notes:

- The decryption functions (DECRYPT_BIT, DECRYPT_BINARY, DECRYPT_CHAR, and DECRYPT_DB) functions return a value that is the result of decrypting encrypted data. The password used for decryption is either the password-string value or the ENCRYPTION PASSWORD value (assigned by the SET ENCRYPTION PASSWORD statement). The decryption functions can only decrypt values that are encrypted using the ENCRYPT_RC2 function. If the encryption facility is not available an error is returned.

DB2 Encryption - Selecting records

- Searching non-encrypted records is business as usual
- Searching through encrypted data can be accomplished in two ways:

1

```
SET ENCRYPTION PASSWORD :enckey1
SELECT name, type FROM ord1
  WHERE DECRYPT_CHAR(cardno) = '3750 999999 1111'
```

- Every column will be decrypted before being compared with the search argument
- Not recommended for performance reasons
- Might be required, when selecting records based on substrings / patterns

```
SELECT name, type FROM ord1 WHERE DECRYPT_CHAR(cardno) LIKE '3750%'
```

2

```
SET ENCRYPTION PASSWORD :enckey1
SELECT name, type FROM ord1
WHERE cardno = ENCRYPT_RC2('3750 999999 1111')
```

- Encrypts search argument and compares encrypted search argument with encrypted values in table columns
- Fastest way to search

Notes:

Selecting records based on encrypted column data can be done in two different ways:

1. You can decrypt each column before comparing the decrypted value with the selection criteria. This way, all columns need to be decrypted. From a performance point of view, this is not the recommended way. However, if you want to select records based on a substring of column data, this is the only option.
2. The faster way of selecting records based on encrypted columns is to encrypt the selection criteria first and then select records based on the encrypted search (selection) criteria. This method does not allow you to select records based on a substring value.

DB2 Encryption
Encryption Handling in SQL Triggers

- SQL triggers or external triggers can be used to automate encryption and decryption tasks
- Before Insert or Before Update triggers can be used for encryption
- If decrypted values are required within triggers, decryption can also be handled

```
CREATE TRIGGER prot_cardno_ins BEFORE INSERT ON ord1
REFERENCING NEW ROW AS x
FOR EACH ROW
BEGIN
  DECLARE passwd VARCHAR(127);
  SET passwd = getPasswdUDF('CARDNO');
  SET x.cardno = ENCRYPT(x.cardno, passwd);
END
CREATE TRIGGER prot_cardno_upd BEFORE UPDATE ON ord1
REFERENCING NEW ROW AS x
FOR EACH ROW
BEGIN
  DECLARE passwd VARCHAR(127);
  SET passwd = getPasswdUDF('CARDNO');
  SET x.cardno = ENCRYPT(x.cardno, passwd);
END
```

Notes:

- A rather smart approach to encrypt data using the DB2 built-in encryption support is using database trigger programs. This method typically does not require application program changes. You can use the BEFORE INSERT and BEFORE UPDATE triggers to encrypt data before they get stored in the DB. It is important to obtain the password dynamically during program execution. Never hardcode passwords in programs. For example, a user defined function can be used to retrieve a password.

DB2 Encryption - Decryption with Views

- SQL statements can be used to decrypt data within applications
- Decryption can also be done in DB Views
- Native programs can open view as logical file
  - Password (key) should (must) be set in user application

```
CREATE VIEW decord2 AS
SELECT DECRYPT_CHAR( cardno ) cardno , name, type FROM ord1
```

```
SQL statement:  SELECT * FROM ord1
CARDNO                                        NAME             TYPE
**úŸŸŸN×Ê×4*±À*®¥wMK*R3*2.**U*·f              Marion           Amex
*ú*ŸŸŸN½fÞ0**Ö**gQ Nu*^úi°*µÃe ¡              Thomas Barlen    Amex
*¦o*ŸŸN]hintkey1ÚäÊ° Fq)ü**á"*ÍÁ®*l@èæRr      Sparky theDog    Visa
*pÑŸŸŸN½_*o¼*ßæ c<Á:.¥ä*´Ù*«3uÑ8              Clarissa         Visa
```

```
SQL statement:  SELECT * FROM decord2
CARDNO                                        NAME             TYPE
4010 8888 77777 1111                          Marion           Amex
4444 4444 4444 4444                           Thomas Barlen    Amex
3750 333333 1111                              Sparky theDog    Visa
1111 1111 1111 1111                           Clarissa         Visa
```

Notes:

- An alternative to decrypting data within application programs provides the use of DB views. As depicted on the previous chart, you can use the DECRYPT_xxx functions when creating a view. In this case, the password must be set in the user application prior to opening the view.

Implementation Planning and Requirements

Implementation Planning

1. Install the prerequisites (more later…)
2. Set up proper access control / object level security
   - Without tight object level security, encryption becomes worthless
   - Access to encryption/decryption processes must be controlled
3. Plan for and set up key management and key store
   - Where to store the key
   - How often will the key be refreshed/changed
   - Protection of the key
4. Decide how and where to implement encryption
   - What data need to be encrypted
   - Evaluate impact on performance
   - How to obtain the key – through a single well protected and controlled service program
   - Who has access to the key
   - Where to implement encryption – application or trigger programs
   - How to handle search/find/lookup operations
5. Backup strategies
6. Initial encryption process / data migration (more later…)

Notes:

- Before you begin using encryption, you need to meet the prerequisites.
- It should be noted at this time, that data encryption is just an additional level of protection. It does not replace the need for proper object level security. You should always set the correct object level permissions first and then start with encryption.
- You also need to properly plan for key generation and management. This is probably the most difficult task in the whole process.
- Decide on how you want to handle encryption and decryption. As discussed in the previous charts, you have the choice of various options, such as Cryptographic Services APIs, CCA APIs, and DB2 built-in encryption. You also need to decide if you implement encryption in the application itself or use trigger programs.
- Another task that deserves special attention is the backup process. You always need to keep the encryption keys for all backed up data. Without the correct encryption key, no decryption is possible.
- Last but not least, you also need to plan for the initial encryption of existing data.

Implementation Prerequisites

- Prereqs
  - –5722-AC3 Cryptographic Access Provider (V5R3, not required in V5R4)

Cryptographic functions	OS/400 - i5/OS	Crypto HW
CCA APIs w/4758	V5R1, V5R2, V5R3, V5R4 incl. opt 35	4758-023
CCA APIs w/4764	V5R3, V5R4 incl. opt 35	4764
Crypto Services APIs	V5R4 V5R3 V5R2* (plus PTFs)	2058 (optional) **
DB2 UDB Encryption	V5R4 V5R3	None

* not all Cryptographic Services APIs are available under V5R2
** not all APIs and algorithms are supported on the 2058

Implementation Requirements and Issues

- Issues that need to be addressed
  - First time encryption
    - how to convert existing clear text data into encrypted data
  - Required changes for the data type and length for columns that will hold encrypted data
    - Decisions have to be made whether current database schemas are changed or auxiliary files will be used
  - Encryption after a key change
    - how to re-encrypt encrypted data after a data encryption key change
  - How to manage KEK changes / re-encryption of KEKs
  - How to handle data import
  - How to handle restore situations
    - when restoring non-encrypted data from older backups
    - when restoring encrypted data that was encrypted under a different data encryption key
  - Are HA environments properly handled (XSM, remote journalling, etc.)

This is certainly not a complete list of issues. You may find many more issues in your own environment

Additional Information and References

- IBM Cryptographic Hardware
  http://www.ibm.com/security/cryptocards/
- Release 2.53 (Windows 2000 and IBM AIX, 4758-002/023), Release 2.54 (IBM i5/OS, 4758-023), and Releases 3.20 and 3.23 (IBM i5/OS, 4764) combined CCA Basic Services manual
  http://www.ibm.com/security/cryptocards/pdfs/bs323mstr.pdf
- IBM Redbook: IBM eServer iSeries Wired Network Security, SG24-6168
  http://www.redbooks.ibm.com/abstracts/sg246168.html?Open
- iSeries Information Center topics:
  ->Programming->APIs->APIs by category->Cryptographic Services
  ->Security->Cryptographic hardware
  ->Database->Reference->SQL Reference (for DB2 encryption information)
  http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm
- iSeries Security Reference, SC41-5302
  http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/books/sc415302.pdf
- DB2 encryption support white paper
  http://www.ibm.com/servers/enable/site/education/ibo/view.html?wp#db2
- Cryptographic Services APIs in InfoCenter
  http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/apis/catcrypt.htm

Additional Information and References

- Cryptographic Services APIs
  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/apis/catcrypt.htm
- Cryptographic Services APIs: A Tutorial
  http://www.iseriesnetwork.com/article.cfm?ID=20312
- Cryptographic Services Key Store
  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/apis/qc3KeyStore.htm
- Cryptographic Services Master Keys
  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/apis/qc3MasterKeys.htm
- Cryptography concepts
  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/rzajc/rzajcconcepts.htm
- PKCS #5: Password-Based Cryptography Standard
  http://www.rsasecurity.com/rsalabs/node.asp?id=2127
- Practical Cryptography
  Wiley Publishing, 2003, ISBN:0-471-22357-3
- Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management
  http://www.ietf.org/rfc/rfc1422.txt?number=1422
- Scenario: Key Management and File Encryption Using the Cryptographic Services APIs
  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/apis/qc3Scenario.htm
- Article on “Cryptographic Services APIs: Key Management” in iSeries iNews magazine.
  (at the time when this presentation was developed, the article was not published yet. Check the iSeries iNews magazine for this article.

Trademarks and Disclaimers

Thank You For Your Attention and Interest

Thomas Barlen
barlen@de.ibm.com