

# **CRYPTO**



# COMPLETE™

ENCRYPTION SUITE FOR SYSTEM i



- Founded in 1994
- Based in Nebraska
- Dedicated to Research and Development
- IBM Advanced Business Partner
- Member of IBM Developer's Program
- Member of PCI Security Standards Council
- Responsive Toll-Free Technical Support



# Linoma's Customers

## Over 3,000 Installations Worldwide

---

BeautiControl Cosmetics

Carolina Biological Supply

Certegy

City of Ketchikan

City of Redding

Consolidated Telephone Companies

CU\*Answers

Discovery Toys

EOG Resources

Fairmount Minerals

Fidelity Express

The Geo Group Inc.

Hermann Sons

Ingram Industries

KOA Campgrounds of America



Korta Payments

Landau Uniforms

Love's Travel Stops & Country Stores

Mid-Continent Group

Muscatine Foods Corporation

Northwest Natural Gas

Oneida Tribe of Indians of WI

Permanent General Agency

Rural Community Insurance Services

Service Insurance Group

Silverleaf Resorts

Slomin's

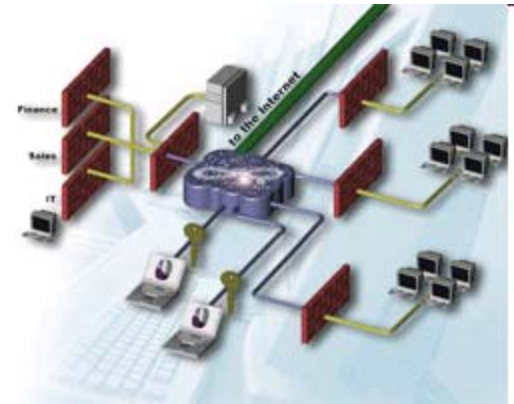
Sturm, Ruger & Company

USA Mobility Wireless

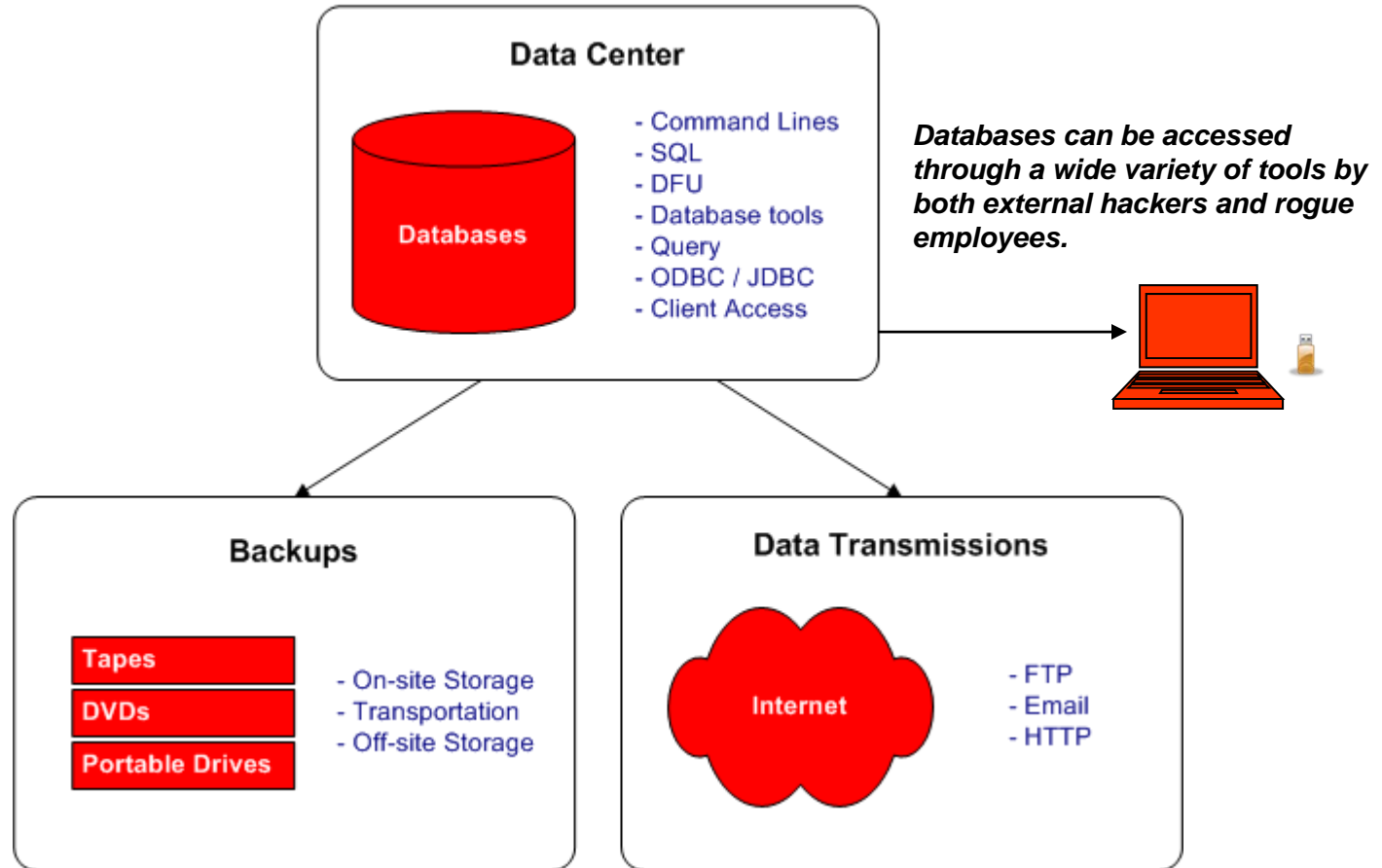
ViaTech Publishing Solutions

# Agenda

- Data risks and regulations
- Key management requirements
- Common pitfalls with encryption
- Crypto Complete for IBM i data encryption
- New features in Crypto Complete 2.20
- Tokenization for centralizing sensitive data
- Controlling access to decrypted values
- Backup encryption
- Audit trails and alerts
- High availability in regards to keys and encryption
- Questions and answers



## Exposed Data



*Backup media often passes through many hands to reach its off-site storage location.*

*Unless otherwise protected, all data transfers travel openly over the Internet and can be monitored or read by others.*





# Costs of Data Breaches

- “Data Loss Study” conducted by Ponemon Institute
- 32,000 lost customer records per breach
- Average cost is over \$200 for each lost record
- \$6.3 million cost per breach
- Costs:
  - Administrative and IT labor
  - Notifications to customers
  - Public relations
  - Regaining trust
  - Lost business

**44 states have enacted legislation  
requiring notification of security breaches  
involving personal information.**


( <http://www.ncsl.org/programs/lis/cip/priv/breachlaws.htm> )

# Data Which Needs a High Level of Protection

- Anything that is confidential to the organization, its employees and its customers
- Credit card numbers    
- Social security numbers
- PIN numbers
- Payroll information (e.g. wages)
- Health-related information (PHI)
- Bank Account numbers
- Driver License numbers
- Financial data
- Trade Secrets (e.g. product formulas)



# Why Should You Protect This Data?

- To comply with regulations:
  - HIPAA
  - Sarbanes Oxley
  - Gramm-Leach-Bliley Act
  - State privacy laws
- To avoid potential penalties and lawsuits
- To comply with PCI Security Standards 
- To avoid bad public relations
- To ensure your continued employment (you don't want to be the one that "takes the fall")

**“A senior database administrator at a subsidiary of Fidelity National Information Services took data belonging to as many as 8.5 million consumers. The stolen data included names, addresses, birth dates, bank account and credit card information, the company said.”**

*(Source: ComputerWorld, July 2007)*

# PCI 1.1 Data Security Standard

- Data Security Standard developed by Payment Card Industry (PCI)
- Latest Standard is 1.2 (released in October 2008)
- View complete text of PCI Data Security Standard at:  
<http://www.pcisecuritystandards.org>
- Excerpt from Standard:



3.4 Render Primary Account Number (PAN), at minimum, unreadable anywhere it is stored (including data on portable digital media, backup media, in logs, and data received from or stored by wireless networks) by using any of the following approaches:

- Strong one-way hash functions (hashed indexes)
- Truncation
- Index tokens and pads (pads must be securely stored)
- Strong cryptography with associated key management processes and procedures.

*The MINIMUM account information that must be rendered unreadable is the PAN.*

## TERMS

A **Key** controls the detailed operations of the encryption and decryption algorithms. A Key is stronger than a typical password and is represented by a series of bits (i.e. 101001011000101...).

**TIP:** The PCI standards can be used as a valuable guide for protecting any sensitive data, not just credit card numbers.

**3.5** Protect encryption keys used for encryption of cardholder data 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 for keys used for encryption of cardholder data, including the following:

**3.6.1** Generation of strong keys

**3.6.2** Secure key distribution

**3.6.3** Secure key storage

**3.6.4** Periodic changing of keys

- As deemed necessary and recommended by the associated application (for example, re-keying); preferably automatically
- At least annually.

**3.6.5** Destruction of old keys

**3.6.6** Split knowledge and establishment of dual control of keys (so that it require two or three 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

**3.6.9** Revocation of old or invalid keys

**3.6.10** Requirement for key custodians to sign a form stating that they understand and accept their key-custodian responsibilities.

**10.0** Track and monitor all access to network resources and cardholder data

*Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.*

# Encryption Basics

- Encryption is the process transforming understandable text into an unintelligible piece of data.
  - **Before:** The quick brown fox jumped over the lazy dog
  - **After:** „Œ \ËKä°BBY ý□\âê·Ñ,C<ÿ^{F+rÀJ[1]Ï(¾Y½i>”®t
- Encryption hides the meaning of the message, but not its existence
- AES is the most popular encryption Cipher.
  - Approved by NIST
  - No known attacks
  - Fast form of Encryption – 6 times faster than Triple DES
  - Can use a 128, 192 or 256 bit symmetric key length

## TERMS

**AES** is the abbreviation for Advanced Encryption Standard. AES utilizes symmetric key cryptology. It provides strong encryption and is approved by the U.S. Government for protecting top secret information.

**Cipher** is a pair of algorithms that perform encryption and decryption.

**NIST** is the abbreviation for National Institute of Standards and Technology. Is a federal technology agency that develops and promotes standards and technology.

### **Quote from National Security Agency (NSA) – June 2003**

"The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths."

# Past Problems with Encryption

- Can be very difficult and time-consuming to implement a custom encryption solution
- IBM's APIs are complex and require a lot of research to implement correctly
- Major application changes often needed to encrypt and decrypt data
- Change database types or lengths to accommodate encrypted data
- Custom key management solution often does not meet PCI requirements:
  - Keys are often not stored securely (often stored in source code or unsecure objects)
  - Lack of controls on who can create and manage keys
  - Difficult to rotate keys without re-encrypting all data
- Audit trails are non-existent or limited
- Programmers know too much about custom solution



# **CRYPTO**



# COMPLETE™

ENCRYPTION SUITE FOR SYSTEM i

# CRYPTO COMPLETE – Overview

- Global Policy Settings (dual control, separation of duties, etc.)
- Integrated Key Management
- Security controls (e.g. who is allowed to set up keys, encrypt data, decrypt data, etc.)
- Rotation of encryption keys without having to re-encrypt existing data
- Strong encryption using AES256, AES192, AES128 or TDES algorithms
- Automated encryption of database fields within local System i database files
  - Utilizes column triggers to automate encryption (without having to change your applications)
  - Encryption of small database fields without requiring field expansion
  - Encryption of both alphanumeric and numeric database fields
- Encryption of files on the Integrated File System (IFS)
- Backup encryption

## Quote from Brad Snapp, City of Owensboro

*"We have found Crypto Complete to be very easy to use. In about an hour, we had our first field encrypted! Crypto Complete gives us the option to automatically encrypt data, which eliminates the need for us to make software changes for encryption."*



# CRYPTO COMPLETE – Overview continued

- Simple program calls and ILE procedures (APIs) for decrypting data within native applications
- Stored procedures and SQL functions for decrypting data through SQL
- Comprehensive audit trails and reporting
- Send message alerts to Email, QSYSOPR, QHST, QAUDJRN...
- Intuitive i5/OS menus and commands with on-line help text

```
CRYPTO                               Main Menu

Select one of the following:

  1. Key Policy and Security Menu      (GO CRYPTO1)
  2. Master Key Menu                  (GO CRYPTO2)
  3. Symmetric Key Menu                (GO CRYPTO3)
  4. Field Encryption Menu             (GO CRYPTO4)
  5. Library/Object/File Encryption Menu (GO CRYPTO5)
  6. Source Examples Menu              (GO CRYPTO6)
 10. Product Information Menu          (GO CRYPTO10)

Selection or command
====>_____

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
F13=Information Assistant   F16=AS/400 main menu
```



# CRYPTO COMPLETE - Key Management Features

- Establish policy settings on how Symmetric Keys can be created and utilized
- Indicate which users can create and manage Symmetric Keys
- Randomly generate strong Symmetric Keys
- Protect Symmetric Keys using Master Encryption Keys
- Organize Symmetric Keys into one or more Key Stores
- Restrict access to Key Stores using i5/OS object authority
- Restrict the retrieval of the actual Symmetric Key values
- Provide separation of duties (i.e. the creator of a Symmetric Key can be restricted from using the Key to encrypt and/or decrypt data)
- Control which users can utilize Symmetric Keys to encrypt and decrypt data



# CRYPTO COMPLETE - Key Hierarchy

## PEK – Product Encryption Key

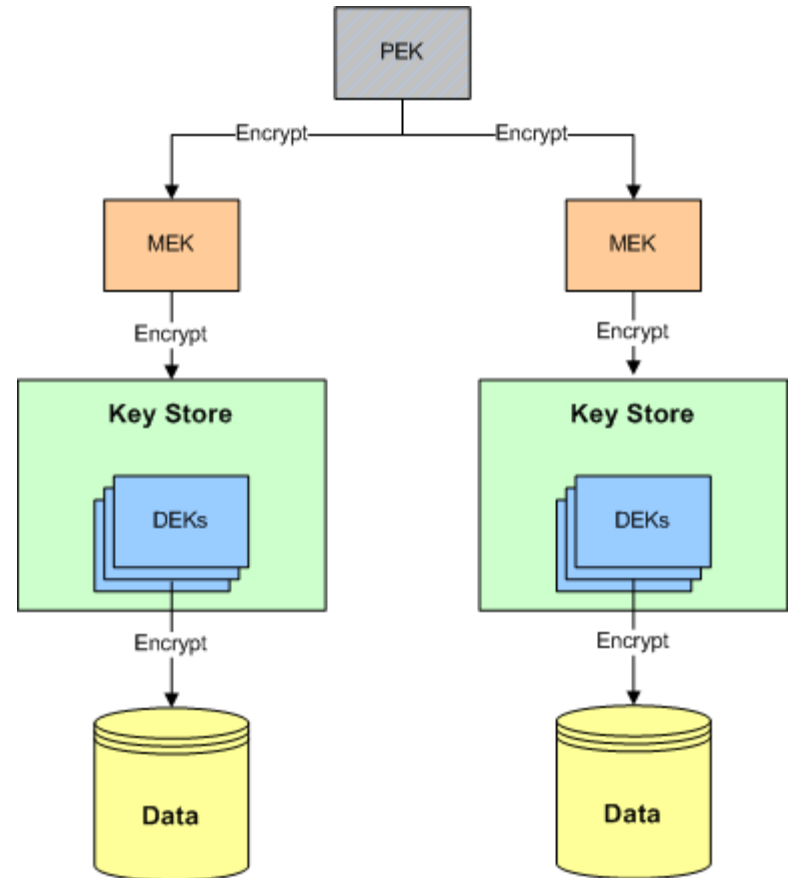
- Quantity: 1
- Used for protecting Master Encryption Keys (MEKs)
- Unique per iSeries serial number
- Only generated in memory when needed (never stored)

## MEK - Master Encryption Keys

- Quantity: 1-8
- Used for protecting Data Encryption Keys (DEKs)
- Generated based on 1-8 passphrases
- Stored in validation list (\*VLDL) object CRVL001

## DEK – Data Encryption Keys

- Quantity: Unlimited
- Used for protecting (encrypting) data
- Can be created 3 ways:
  - 1) Random
  - 2) Generated based on passphrase
  - 3) Manually entered
- DEKs are held in Key Stores
- Key Stores are IBM Validation List (\*VLDL) objects



## CRYPTO COMPLETE – New in 2.20

- Faster Backup Encryption (twice the speed as prior release)
- Much faster Mass Encryption of Database Fields (for initial activation)
- Supports tokenization and centralization of sensitive data
- HTTPS interfaces to support diverse systems (e.g. Windows, Linux, AIX, etc.)
- Log message routing to SYSLOG and Protegrity Enterprise Security Administrator
- Other minor enhancements and fixes



# CRYPTO COMPLETE – Field Encryption

- Specify fields to encrypt within Field Encryption Registry
- Supports both local and remote fields
- Encrypted values can be stored within existing database field or external file

```
7/16/07          Work with Field Encryption Registry          BLUEBBE
22:04:19                                               CRRM040          D2

Type options, press Enter.
  2=Change   4=Remove   5=Display   7=Activate   8=Deactivate
 10=Change Key 12=Display Key History

Opt  Field identifier          Database field          Status
---  -
---  BANK_ACCOUNT              BANKNO                  *ACTIVE
---  BIRTH_DATE                 BTHDATE                *INACTIVE
---  CREDIT_CARD                CCNO                    *ACTIVE
---  SOCIAL_SECURITY_NBR        SOCIAL                  *PROCESS
---  SQL_SERVER_CREDIT_CARD     *REMOTE                 *ACTIVE

F3=Exit  F5=Refresh  F6=Add  F11=View2  F12=Cancel
```

## **Demo of Local Field Encryption**

# Tokenization – Introduction

- Keys are managed and secured on a single server
- Encrypted values are indexed and stored in a central location
- Data is protected using strong AES encryption
- Supports both alpha and numeric database fields
- Based on central access controls, returns either fully decrypted or masked values
- Can support diverse platforms (e.g. System i, Windows, Linux, etc.)
- Helps limit the scope of audits



# Tokenization - Diagram

## Token Server

( IBM System i )

### Crypto Complete™



Keys



Encrypted Values  
(Indexed by Token Id)

Key Management

Security Controls

Encryption Functions

Token and Data Management

Interfaces

Audit Trails

Example Token Database:

Token Id	Encrypted Value	User	Timestamp
5483795	XXXXXXXXXXXXXXXXXX	bluebbe	10/2/09 10:30:12
6787872	XXXXXXXXXXXXXXXXXX	jsmith	10/3/09 11:31:22
7026592	XXXXXXXXXXXXXXXXXX	sperry	10/6/09 09:35:33
8534620	XXXXXXXXXXXXXXXXXX	mwiser	10/6/09 13:37:65
9682028	XXXXXXXXXXXXXXXXXX	bpickie	10/8/09 15:38:76

### Storage Process

Data (to encrypt/store), Field Id, User

Token Id

### Retrieval Process

Token Id, Field Id, User

Decrypted Full Value or Masked Value

## Other Servers

( System i, Windows, Linux, Unix, Mainframe, etc.)

### Customer Applications

(RPG, COBOL, JAVA, .NET, PHP, C...)

### Crypto Complete™ Token APIs

- HTTPS
- Stored Procedures
- Data Queues



Customer Data

Example Customer Database:

Customer#	Name	Credit Card
17372	ABC Company	5483795
23525	Silver Street Shop	6787872
33323	Gift Niche	7026592
45675	Marcy Mutuary	8534620
53875	Lee Sapp Ford	9682028

Token Ids are stored in existing database fields

# CRYPTO COMPLETE – Storage of Encrypted Values

- Encrypted values are stored in a separate physical file (one for each field)
- Approach used in tokenization for remote fields. Can also be used for local fields.
- Token file layout:

Field	Example value	Optional
Field identifier	CREDIT_CARD	
Token (index) number	7	
Key id	2	
Last updated by user	BILL	
Last updated time	2008-07-10-18.09.39.375000	
Last retrieved by user	MARY	Yes
Last retrieved time	2008-07-15-01.22.32.567000	Yes
Record hash	.....	Yes
Encrypted value	.....	

- For above example, original database field will contain index number of 7
- Allows rotating keys at any time without having to re-encrypt data

## Demo of Tokenization

# CRYPTO COMPLETE – Token HTTP APIs

- Get Connection:

```
HTTP_GetConnection(Host  
                  :Port  
                  :SSL  
                  :ApplicationId  
                  :UserId  
                  :Password  
                  :MsgId  
                  :MsgText);
```

Required inputs. Host and port of token server.

- Insert value:

```
HTTP_InsEncFld('CCFIELD'  
              :CreditCardValue  
              :LogCmt  
              :TokenId  
              :MsgId  
              :MsgText);
```

Required inputs. Field identifier and data.

Output. Store token id in existing database field.

- Retrieve decrypted value:

```
HTTP_GetEncFld('CCFIELD'  
              :TokenId  
              :LogCmt  
              :CreditCardValue  
              :MsgId  
              :MsgText);
```

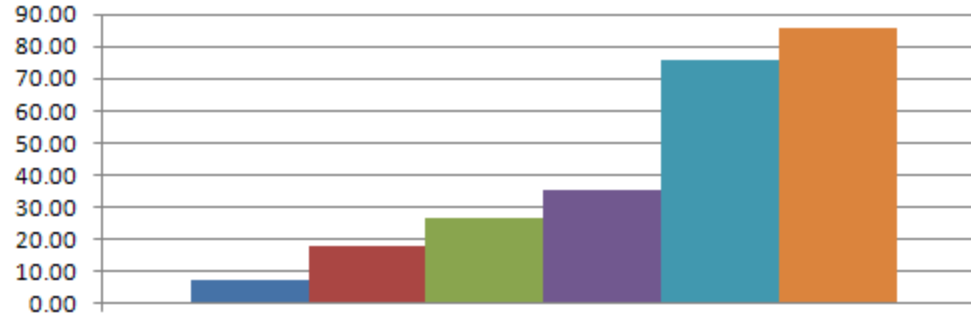
Required inputs. Field identifier and token id.

Authorized output. full value, masked value or no value

# Performance Tests on IBM i 520

Insert and Encrypt:

	Seconds per 10,000 Records	Seconds per Record
<b>Local Encryption:</b>		
Local- Store in existing field	7.00	0.00070
Local- Store in external file	17.67	0.00177
<b>Tokenized from Remote System:</b>		
Remote IBM i (Data Queue)	26.33	0.00263
Remote from Java on Intel (Stored Procedure)	35.00	0.00350
Remote IBM i (HTTPS)	76.00	0.00760
Remote from Java on Intel (HTTPS)	85.67	0.00857

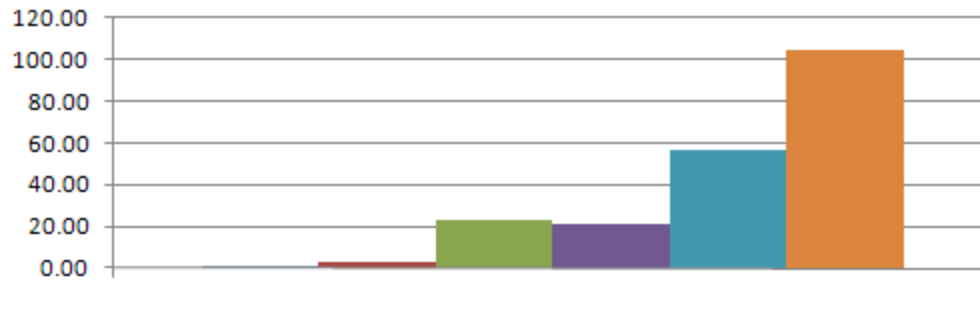


- Local- Store in existing field
- Local- Store in external file
- Remote IBM i (Data Queue)
- Remote from Java on Intel (Stored Procedure)
- Remote IBM i (HTTPS)
- Remote from Java on Intel (HTTPS)

# Performance Tests on IBM i 520

Retrieval:

	Seconds per 10,000 Records	Seconds per Record
<b>Local Encryption:</b>		
Local - Stored in existing field	1.00	0.00010
Local - Stored in external file	2.50	0.00025
<b>Tokenized from Remote System:</b>		
Remote from another IBM i (Data Queue)	23.33	0.00233
Remote from Java on Intel (Stored Procedure)	20.67	0.00207
Remote from another IBM i (HTTPS)	57.00	0.00570
Remote from Java on Intel (HTTPS)	104.67	0.01047



# Tokenization – General Steps

## On the Token Server:

1. Create Data Encryption Keys (DEKs)
2. Set up authorization lists to control access to full and masked values
3. Register and activate the fields
4. Set up connectivity options (HTTPS, Stored Proc or Data Queue)

## On the Remote Servers:

1. Mass encrypt/tokenize existing values with Insert API
2. Modify applications on remote systems to call APIs on Inserts, Updates and Gets

# Tokenization – Considerations

- Authentication between remote and token servers (e.g. certificates for HTTPS)
- Performance considerations
- Field length must be long enough to hold token number
- High availability planning for the token server and communications channel

# CRYPTO COMPLETE – Controlling Access to Decrypted Values

- Secure Key Stores using i5/OS object authority (*If the user does not have rights to the Key Store containing the Key, then they cannot encrypt or decrypt data with that Key*)

**KEYSTORE1**

```

Key . . . . . CREDIT_CARD_KEY
Encryption allowed . *YES
Decryption allowed . *NO

Key . . . . . SSNO_KEY
Encryption allowed . *YES
Decryption allowed . *NO
    
```

Object Authorities

User	Group	Object Authority
*PUBLIC		*EXCLUDE
	DATAENTRY	*USE
	HR	*USE
	MANAGERS	*USE

**KEYSTORE2**

```

Key . . . . . CREDIT_CARD_KEY
Encryption allowed . *NO
Decryption allowed . *YES

Key . . . . . SSNO_KEY
Encryption allowed . *NO
Decryption allowed . *YES
    
```

Object Authorities

User	Group	Object Authority
*PUBLIC		*EXCLUDE
	MANAGERS	*USE

- Specify authorization lists (AUTL) for the field (*controls if the user has access to the full decrypted value, masked value or no value*)

**Field Registry**

```

Field . . . . . CCNO
Auth. list for Full Value . . CCFULL
Auth. list for Masked Value . CCMASKED
    
```

# CRYPTO COMPLETE – Backup Encryption

- Encrypts and saves iSeries libraries, objects and files
- Target to disk, tape and other supported media devices
- Choose between AES128, AES192 and AES256 encryption
- Supports key-based and password-based protection
- Can be integrated into BRMS
- Native i5/OS commands:
  - Encrypt Library (ENCSAVLIB)
  - Decrypt Library (DECRSTLIB)
  - Encrypt Object (ENCSAVOBJ)
  - Decrypt Object (DECRSTOBJ)
  - Encrypt Save File (ENCSAVF)
  - Decrypt Save File (DECSAVF)
  - Encrypt File (ENCFIL)
  - Decrypt File (DECFIL)



## CRYPTO COMPLETE - Example commands

```
/* Save Payroll Library */
ENCSAVLIB  LIB(PAYROLL) DEV(TAP01) VOL(*MOUNTED) +
           SEQNBR(*END) ALGORITHM(*AES256) USEKEYPAS(*KEY) +
           KEYLABEL(BACKUPKEY) KEYSTR(KEYSTORES/BACKUPSTR)

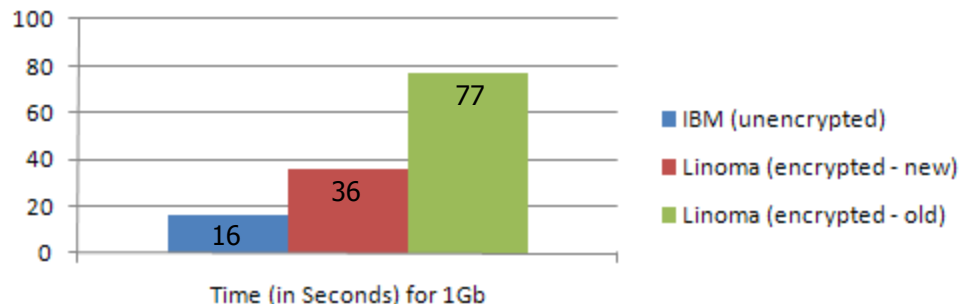
/* Save Order Files */
ENCSAVOBJ  OBJ(ORDERHDR ORDERDTL) LIB(OELIB) +
           OBJTYPE(*FILE) DEV(TAP01) VOL(*MOUNTED)+
           ALGORITHM(*AES256) USEKEYPAS(*KEY) +
           KEYLABEL(BACKUPKEY) KEYSTR(KEYSTORES/BACKUPSTR)
```

```
/* Restore Payroll Library */
DECRSTLIB  SAVLIB(PAYROLL) DEV(TAP01) VOL(*MOUNTED) +
           USEKEYPAS(*KEY) KEYLABEL(*AUTO)

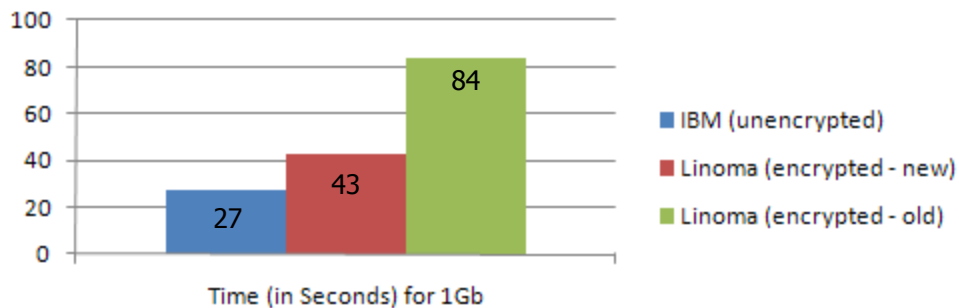
/* Restore Order Files */
DECRSTOBJ  OBJ(ORDERHDR ORDERDTL) SAVLIB(OELIB) +
           OBJTYPE(*FILE) DEV(TAP01) VOL(*MOUNTED) +
           USEKEYPAS(*KEY) KEYLABEL(*AUTO)
```

# CRYPTO COMPLETE – Backup Encryption

## Backup Speeds:



## Restore Speeds:

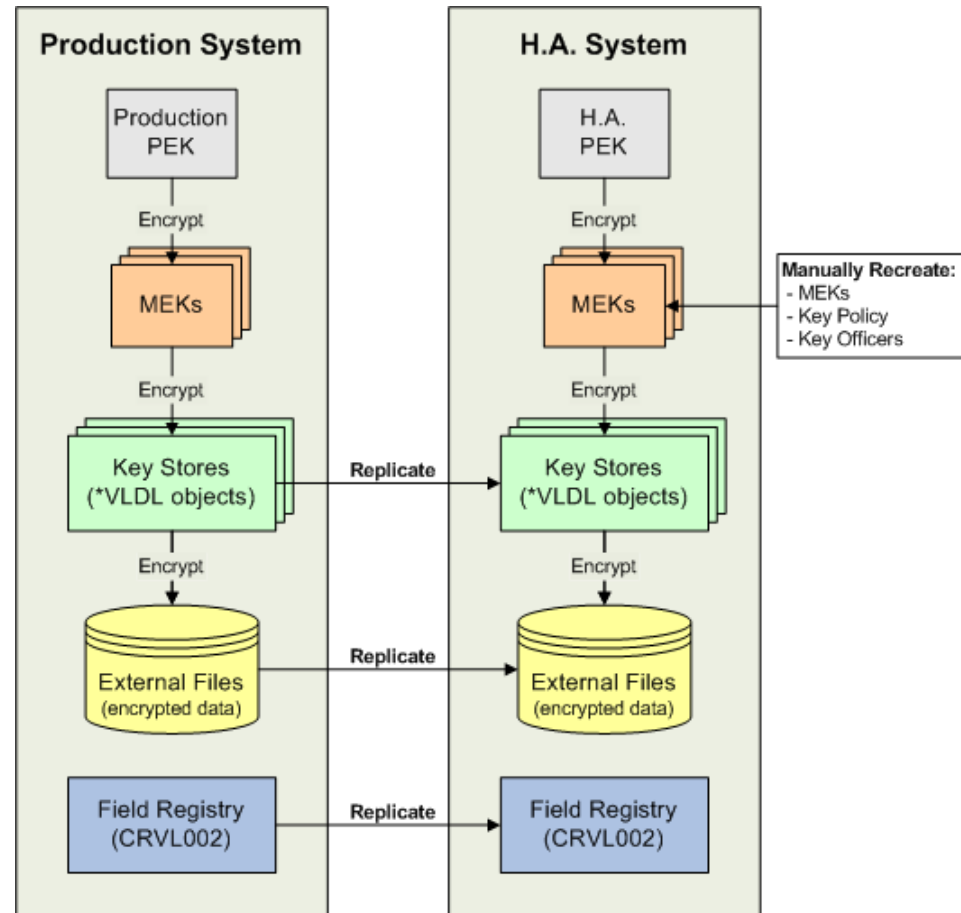


## Recommendations to minimize backup window:

- Only encrypt libraries or objects which contain sensitive data
- Save to Save Files first, and then backup Save Files using ENCSAVF command (if enough disk space available)

# CRYPTO COMPLETE – High Availability Environment

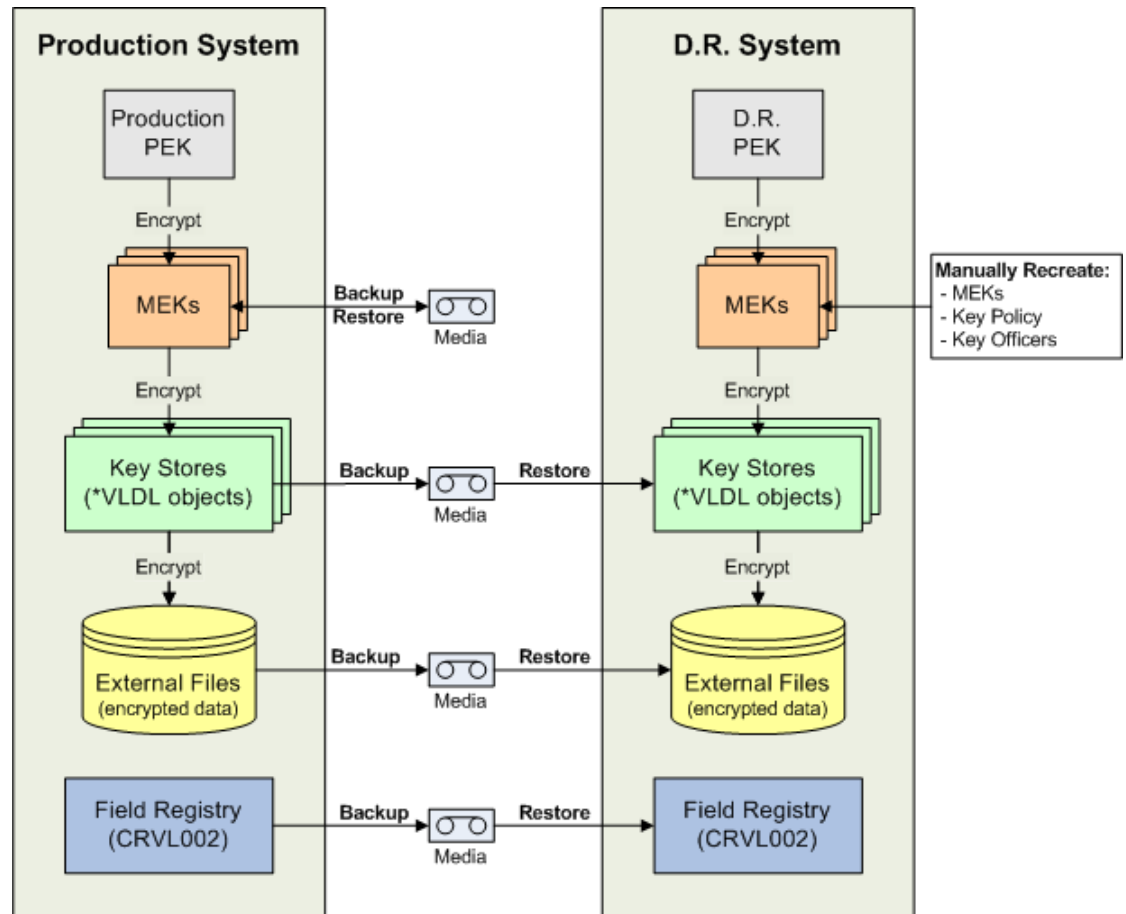
- PEK is different per System
- Recreate Master Keys on H.A. system using same passphrases as used on Production system
- Replicate Key Stores
- Replicate any external files containing encrypted data
- Replicate Field Registry



PEK – Product Encryption Key  
MEK – Master Encryption Key  
DEK – Data Encryption Key

# CRYPTO COMPLETE – Backup and Recovery

- PEK is different per System
- Production Backup:
  - Master Keys
  - Key Stores
  - External files
  - Field Registry
- If recovery on Production system:  
Restore Master Keys from media
- If recovery on D.R. system:  
Recreate Master Keys on D.R. system using same passphrases as used on Production system
- Restore:
  - Key Stores
  - External files
  - Field Registry



PEK – Product Encryption Key  
MEK – Master Encryption Key  
DEK – Data Encryption Key

# CRYPTO COMPLETE – Audit Trails

- Comprehensive audit trails
- Stored in secure IBM Journal
- Types of activity audited:
  - When any Key Policy settings are changed
  - When Key Officers are added, changed or removed
  - When Master Encryption Keys (MEKs) are loaded or set
  - When Key Stores are created or translated
  - When Data Encryption Keys (DEKs) are created, changed or deleted
  - When Field Encryption Registry entries are added, changed, removed, activated or deactivated
  - When any functions are denied due to improper authority
  - When data is encrypted or decrypted with a key that requires logging of those events
  - When data cannot be encrypted or decrypted due to errors (i.e. invalid key label specified)
- Generate reports based on:
  - User
  - Date range
  - Audit type

## CRYPTO COMPLETE – SQL Triggers (for local fields)

- Product can create SQL Triggers used to trap inserts, updates and deletes
- Trigger will call API to perform the encryption
- Update trigger is column based (only runs when that particular field changes)
- Very good performance
- SQL triggers are saved with the database file (unlike an external trigger)

# CRYPTO COMPLETE – Retrieve encrypted value (for local fields)

- Pass in field identifier and encrypted value
- Get back the authorized value (full value, masked value, or no value)
- Example of calling ILE procedure to retrieve decrypted value :

```
DecFldAuth ( 'Credit_Card'  
            :EncryptedValue  
            :LogCmt  
            :CreditCardValue  
            :MsgId  
            :MsgText ) ;
```

- Also include APIs that can be called with traditional CALL statement
- SQL functions and Stored procedures also available

```
SELECT CustNo,  
       F_DecFldAuth( 'Credit_Card', CreditCard)as Decrypted_CreditCard  
FROM OrderFile  
WHERE CustId = 12345
```

# CRYPTO COMPLETE – SUMMARY

- Free 30 day trial available for download
- Installs as a licensed program – Uses only 75 Mb of disk
- Most customers can install and start encrypting data in less than a couple hours
- Comprehensive easy-to-read manual
- On-line help text
- Evaluate with test data in your own environment

## Customer Testimonial

“There are not a lot of software products that impress me, but I have to say that I really like the way Crypto Complete works. It was easy to implement and allowed us to meet all the requirements for securing our data to get PCI compliant.”

- Will Crowe, Love's Travel Stops and Country Stores



# How to contact us



L I N O M A

S O F T W A R E

REVOLUTIONARY SOLUTIONS FOR YOUR WORLD

**Web site:** [www.linomasoftware.com](http://www.linomasoftware.com)  
**E-mail:** [sales@linomasoftware.com](mailto:sales@linomasoftware.com)

**Toll-free:** 1-800-949-4696  
**Direct:** (402) 944-4242  
**Fax:** (402) 944-4243

**Address:** 1409 Silver Street  
Ashland, NE 68003 USA