

DB-9 Data Encryption Made Easy  
Michael G. Solomon

**DB-9: Data Encryption Made Easy**

How to properly manage encryption keys

**Michael G. Solomon**  
CISSP GSEC PMP CISM  
Solomon Consulting Inc.

**Exchange** PROGRESS®

**DB-9: Data encryption made easy**

- 1 Who needs encryption?
- 2 Encryption 101
- 3 Common encryption pitfalls
- 4 Overcoming encryption pitfalls

DB-9 Encryption Made Easy

**Exchange** PROGRESS®

## Who needs encryption?

Layered Defense

Legislation

Vendor compliance

Offline safety

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Encryption 101

- What is encryption?
  - Scrambling data in a special way
  - Only the intended recipient can read it
  - Most common use – data confidentiality
  - Can also ensure data integrity
- Important terms
  - Plaintext – **unencrypted message**
  - Ciphertext – **encrypted message**
  - Algorithm – **steps to encrypt and decrypt**

DB-9 Encryption Made Easy

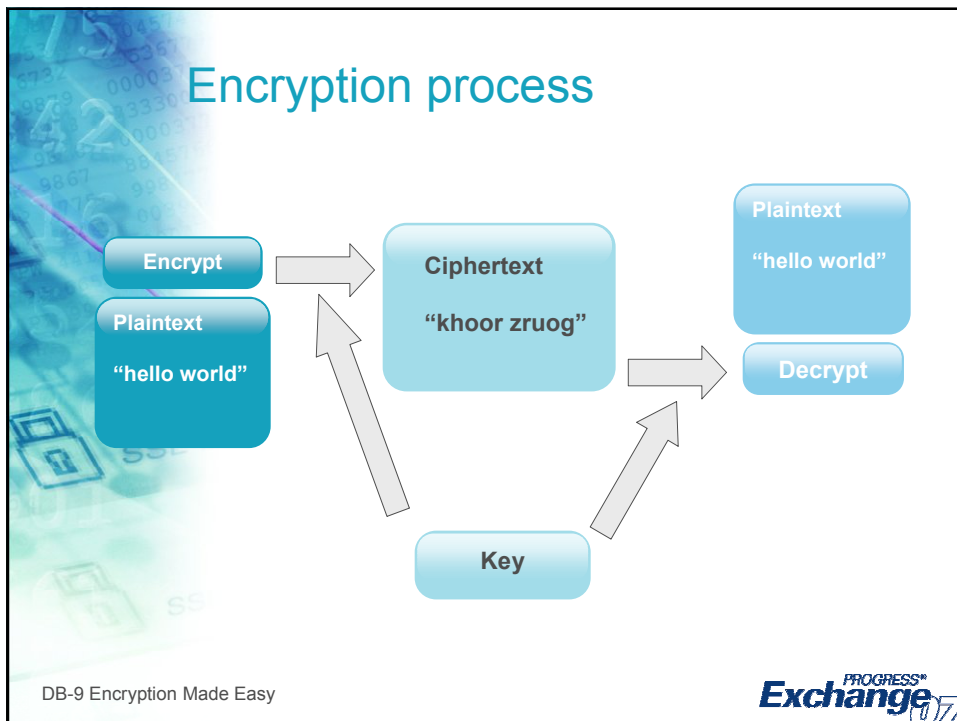

PROGRESS®  
**Exchange**

## Encryption process

- Plaintext – “hello world”
- Algorithm – Caesar cipher (ROT3)
  - Transpose (shift) each character 3 positions to the right
- Ciphertext – “khood zruog”

Original	Encrypted
a	d
b	e
c	f
d	g
e	h
etc.	etc.

DB-9 Encryption Made Easy



## Encryption Syntax in OpenEdge®

```
DEF VAR bKey AS RAW.  
DEF VAR cipherText AS RAW.  
DEF VAR algo AS CHAR INIT  
    "AES_CBC_128".  
  
SSL bKey = GENERATE-RANDOM-KEY.  
  
cipherText = ENCRYPT( "hello  
world", bKey, ?, algo).
```

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Decryption process

- Opposite of encryption
  - Transpose (shift) each character three positions to the *left*
- The “key” - number of positions to shift
  - In our example: 3
  - Must be agreed upon
- Same key used to encrypt and decrypt
  - In this case
  - Some encryption uses different keys

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

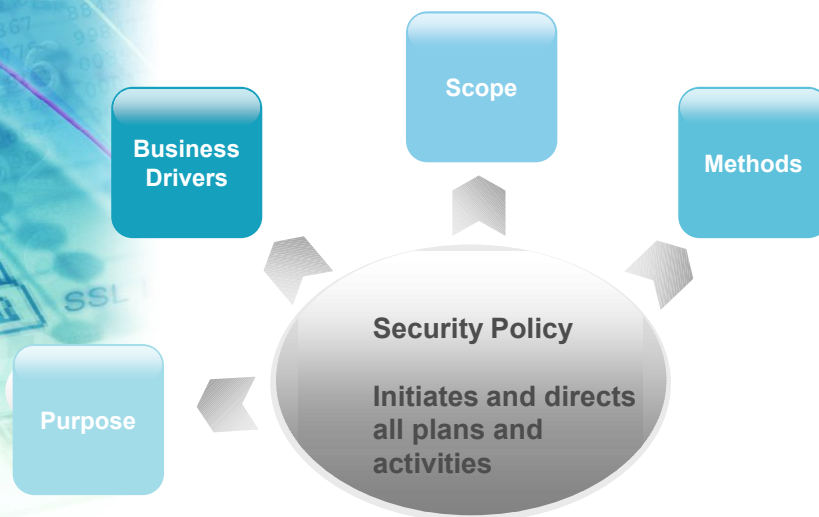
## Decryption Syntax in OpenEdge

```
/* continued from prev slide */  
  
DEF VAR plainText AS RAW.  
  
plainText = DECRYPT( cipherText,  
                    bKey, ?, algo).  
  
MESSAGE GET-STRING(plainText,1)  
VIEW-AS ALERT-BOX.
```

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## The Real First Step

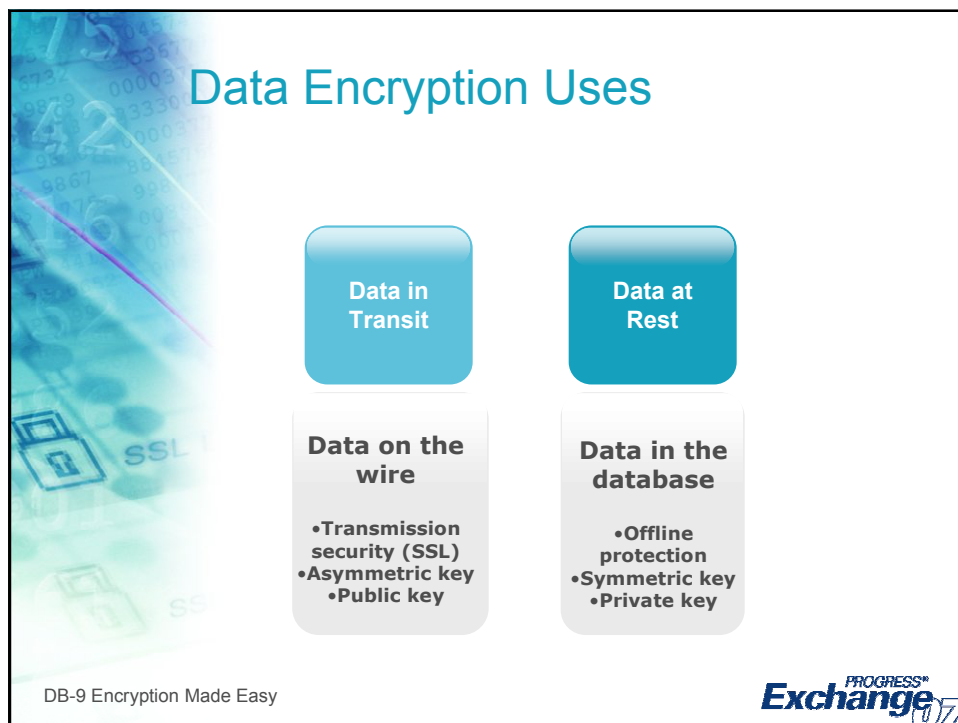
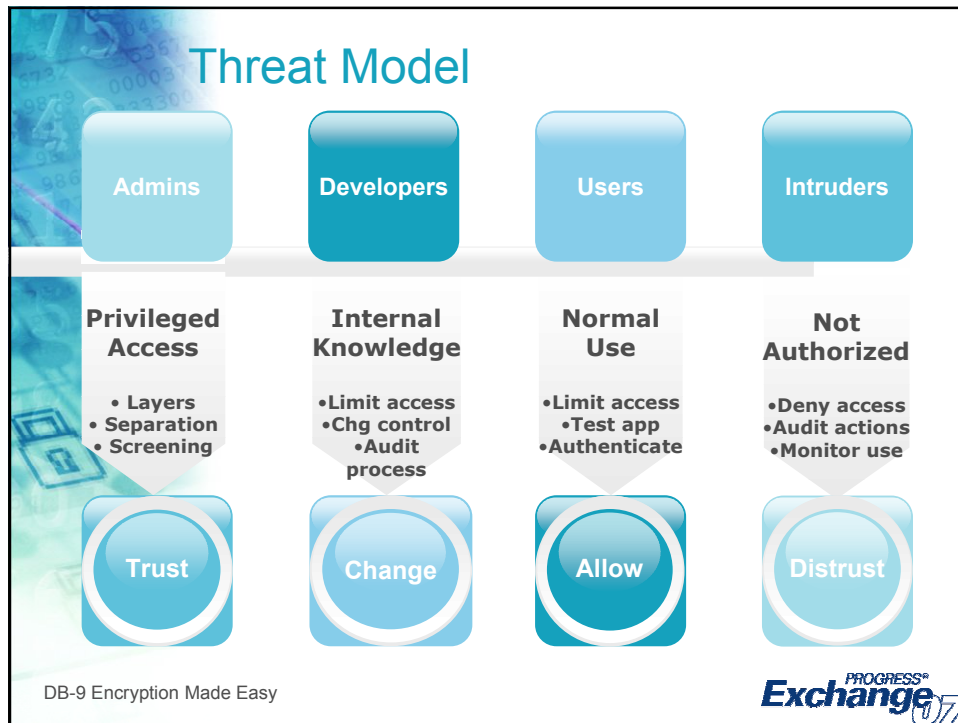


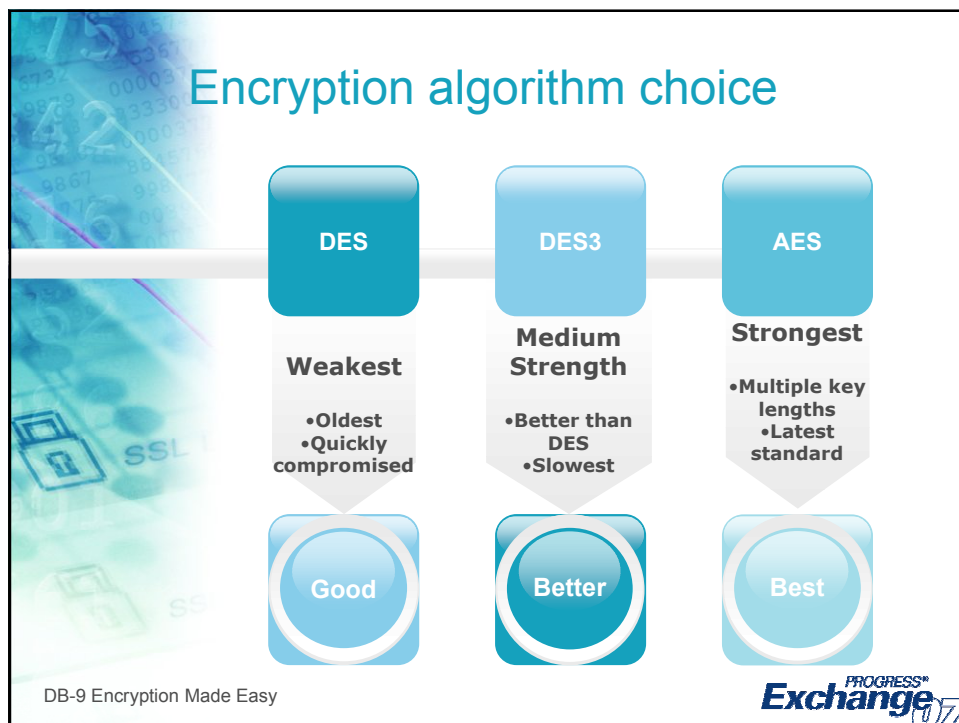
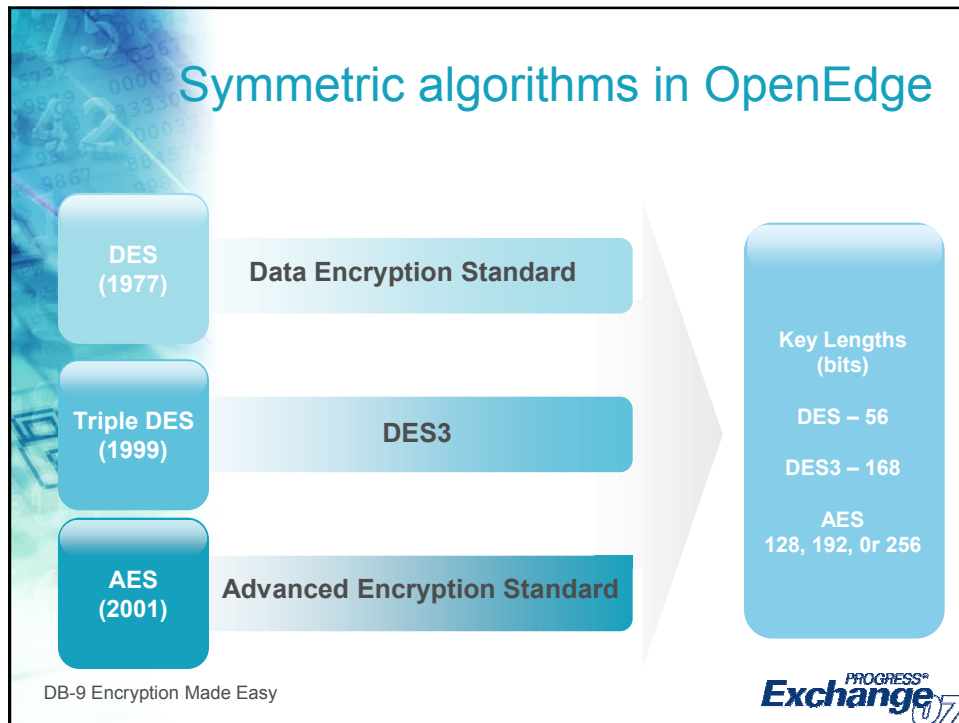
DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**


# DB-9 Data Encryption Made Easy

Michael G. Solomon





## Key Analogy

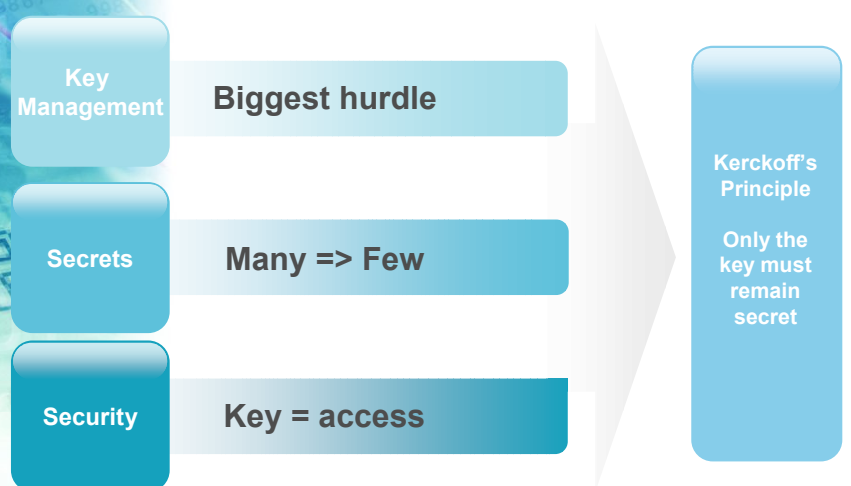


- Consider a hotel
- Employee access to rooms
- Where are keys stored
- Guests check out
- How are keys protected

DB-9 Encryption Made Easy

**Exchange** PROGRESS®

## Protect the keys!



**Key Management** — **Biggest hurdle**

**Secrets** — **Many => Few**

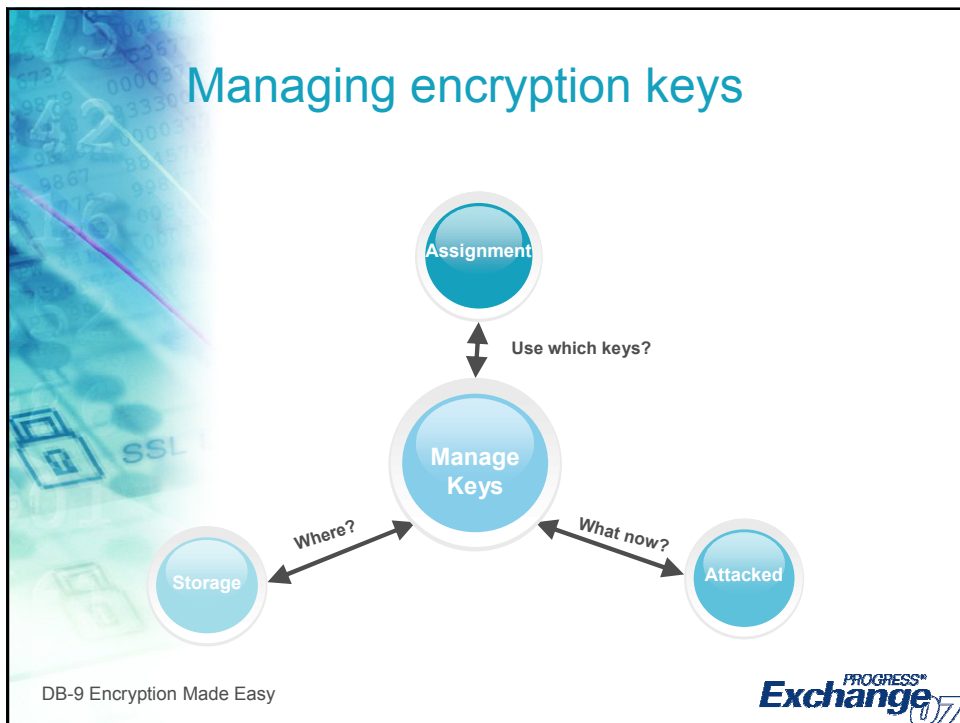
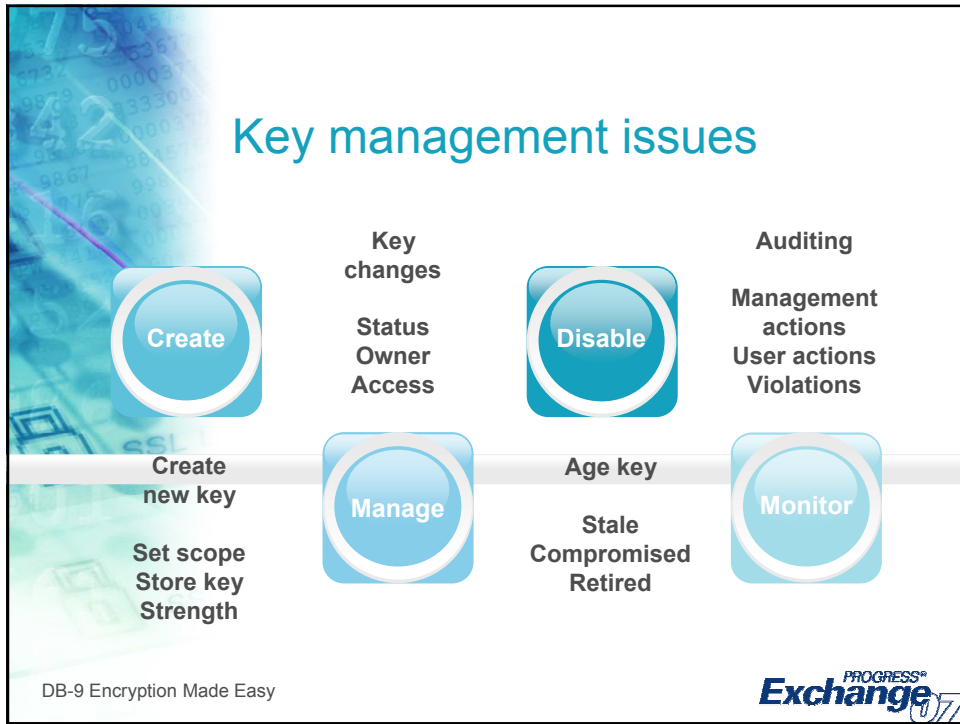
**Security** — **Key = access**

**Kerckhoff's Principle**  
Only the key must remain secret

DB-9 Encryption Made Easy

**Exchange** PROGRESS®

DB-9 Data Encryption Made Easy  
 Michael G. Solomon



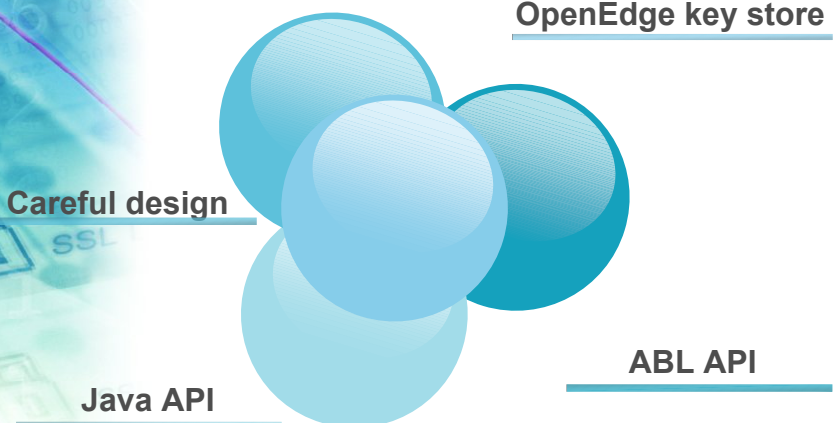
## Where Can I Store Keys?

In the code	Encrypted password	Key store
<b>Bad option</b> <ul style="list-style-type: none"><li>• Easy to hack</li><li>• Hard to change</li><li>• No mgmt</li></ul>	<b>Still not good</b> <ul style="list-style-type: none"><li>• Eggs in one basket</li><li>• Human weakness</li><li>• Little mgmt</li></ul>	<b>Best method</b> <ul style="list-style-type: none"><li>• Flexible</li><li>• Secure</li><li>• Easy to manage</li></ul>

DB-9 Encryption Made Easy



## A key management solution




Careful design

OpenEdge key store

Java API

ABL API

DB-9 Encryption Made Easy



## Basic key store schema

- Table: pks\_keyEncryptingKey
  - pks\_kekId (INT)
  - pks\_keyData (RAW)
  - pks\_actDate (DATETIME)
  - INDEX on actDate
- Table: pks\_keyStore
  - pks\_keyId (INT)
  - pks\_keyData (RAW)
  - pks\_kekId (INT)
  - INDEX on keyId

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

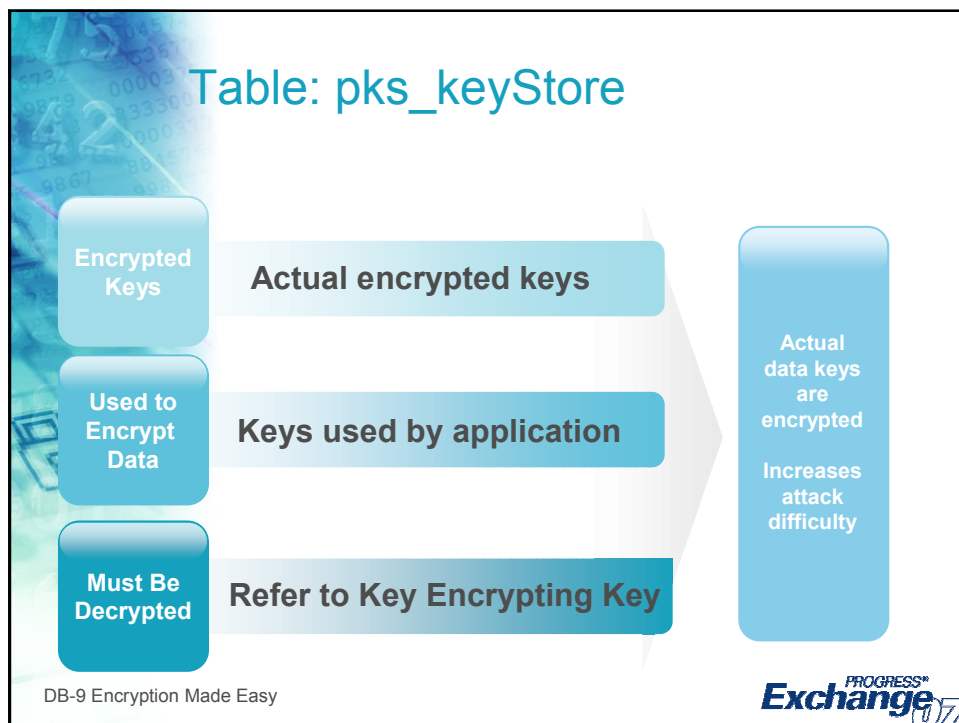
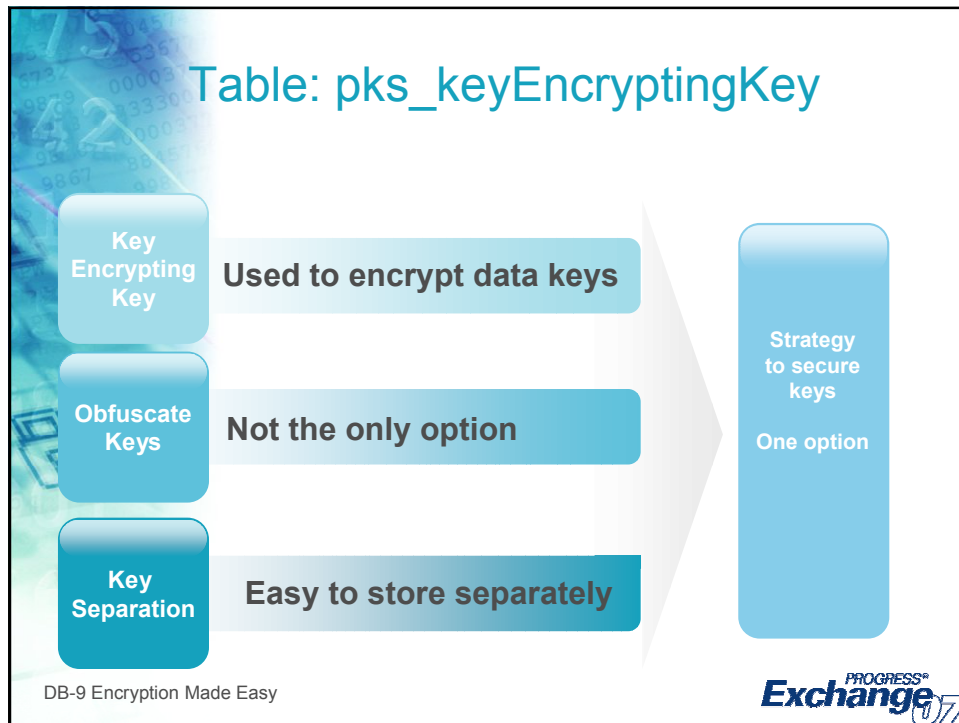
## Basic key store schema

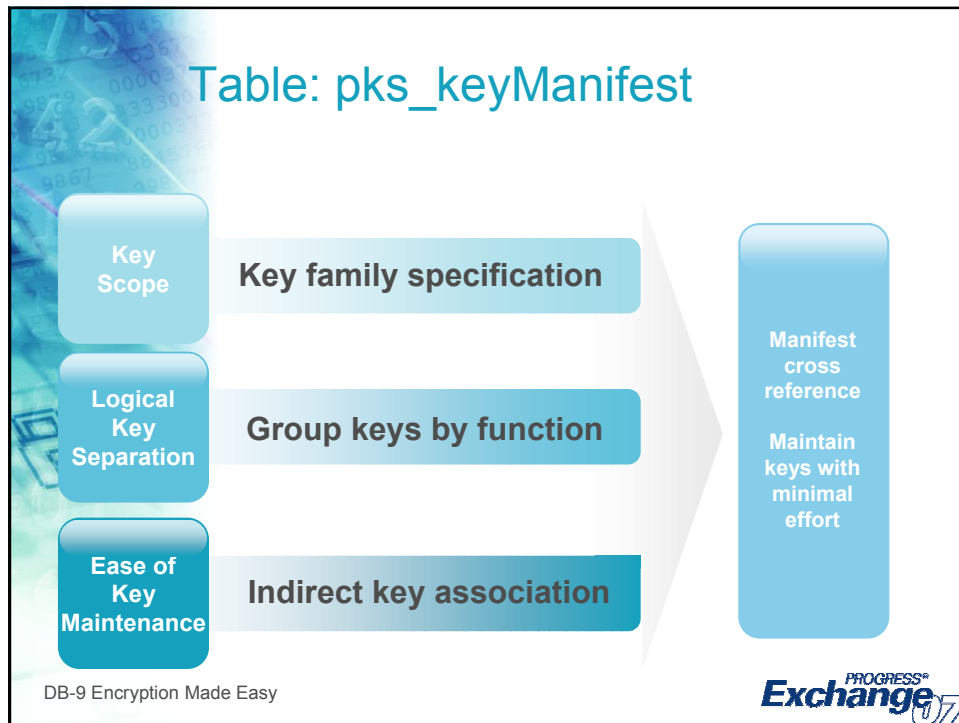
- Table: pks\_keyManifest
  - pks\_aliasId (INT)
  - pks\_keyAlias (CHAR)
  - pks\_keyFamily (CHAR)
  - pks\_keyId (INT)
  - pks\_keyActDate (DATETIME)
  - pks\_keyStatus (CHAR)
  - INDEX on aliasId
  - INDEX on keyActDate

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

DB-9 Data Encryption Made Easy  
Michael G. Solomon





## Code sample: generateKey()

- Generate a new key

```
def var rawKek as raw no-undo.  
  
rawKek = generate-random-key.  
create pks_keyEncryptingKey.  
assign  
    pks_keyEncryptingKey.keyData = rawKek  
    pks_keyEncryptingKey.actDate = now.  
  
return pks_keyEncryptingKeys.kekId.
```

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Code sample: retireKey()

- Mark an active key as retired
  - Still available for decryption
  - Not available for encryption

```
DEFINE INPUT PARAMETER pAliasId  
    AS INTEGER NO-UNDO.  
  
FIND pks_keyManifest WHERE aliasId  
    = pAliasId EXCLUSIVE-LOCK NO-  
    ERROR.  
  
ASSIGN keyStatus = "RETIRED".
```

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Code sample: encryptData()

- Use current key to encrypt data

```
DEFINE INPUT PARAMETER pAlias AS CHAR.  
DEFINE INPUT PARAMETER pPlainText AS CHAR.  
DEFINE INPUT PARAMETER pAlgo AS CHAR.  
DEFINE OUTPUT PARAMETER pCipherText AS RAW.  
DEFINE OUTPUT PARAMETER pAliasId AS INT.  
  
FIND FIRST pks_keyManifest WHERE  
    pAlias = pks_keyManifest.alias AND  
    pks_keyManifest.keyActDate GE NOW NO-LOCK.  
  
rawKey = decryptKey(pks_keyManifest.keyId).  
pCipherText = ENCRYPT( pPlainText, rawKey, ?,  
    pAlgo).  
pAliasId = pks_keyManifest.aliasId.
```

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Code sample: decryptData()

- Look up key and decrypt data

```
DEFINE INPUT PARAMETER pAliasId AS INT.  
DEFINE INPUT PARAMETER pCipherText AS RAW.  
DEFINE INPUT PARAMETER pAlgo AS CHAR.  
DEFINE OUTPUT PARAMETER pPlainText AS CHAR.  
  
FIND pks_keyManifest WHERE  
    pAliasId = pks_keyManifest.aliasId NO-LOCK.  
  
rawKey = decryptKey(pks_keyManifest.keyId).  
pPlainText = DECRYPT( pCipherText, rawKey, ?,  
    pAlgo).
```

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Code sample: updateEncData()

- Update encrypted data when key status changes
- Combines previous two concepts

```
decryptKey(aliasId, cipherText, algo,  
          OUTPUT plainText).  
encryptKey(alias, plainText, algo,  
          OUTPUT pCipherText, OUTPUT aliasId ).
```

## Putting it all together

- A key's life demo
  - generateKey()
  - updateKey()
  - encryptData()
  - decryptData()
  - generateKey() – a new key
  - retireKey() – the old key
  - decryptData()
  - encryptData()
  - updateEncData()

## Summary

- Plan your key management processes up front
  - **Plan for entire key life cycles**
- Restrict access to key store
- Extend the sample code to include
  - **Multiple algorithms**
  - **Multiple key lengths**

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

## Need more information?

- Solomon Consulting Inc.
  - **OpenEdge® security specialist**
  - **Database and application security services**
    - Vulnerability assessment and remediation
    - Secure development best practices
    - Database lockdown
    - Regulatory compliance
  - **ProKeyStore – OpenEdge® encryption key management system**
    - <http://www.solomonconsulting.com>
- SANS Reading Room
  - [http://www.sans.org/reading\\_room](http://www.sans.org/reading_room)

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

DB-9 Data Encryption Made Easy  
Michael G. Solomon

## Relevant Sessions

- COMP-1 – Securing your web against hackers
- DEV-4 – OpenEdge in an LDAP World
- COMP-7 – Securing Your Swiss Cheese Environment
- DB-19 – OpenEdge Authentication without using the \_User table
- COMP-15 – Disaster Recovery – Be Scared, Be Very Scared

DB-9 Encryption Made Easy

PROGRESS®  
**Exchange**

Thank You!

[www.solomonconsulting.com](http://www.solomonconsulting.com)

PROGRESS®  
**Exchange**

Progress Exchange 2007  
10 – 13 June, 2007, Phoenix, AZ USA